

Supplementary Data 1: Comparison of normalization using real data

Catalina Vallejos, Davide Risso, Antonio Scialdone, Sandrine Dudoit and John Marioni

Contents

Introduction	2
Dataset 1	2
Normalization and gene filtering	2
Comparing normalization strategies	3
Highly variable genes detection	5
Dataset 2	8
Normalization and gene filtering	9
Comparing normalization strategies	9
Highly variable genes detection	11
Dataset 3	14
Normalization and gene filtering	14
Comparing normalization strategies	15
Highly variable genes detection	17
Dataset 4	20
Subpopulation 1: astrocytes-ependymal	21
Subpopulation 2: oligodendrocytes	26
Subpopulation 3: microglia	31
Appendix	36
Comparison of global properties of the datasets	36
Function definitions	38
References	42

Introduction

This document is provided as supplementary material for the article

Normalizing single-cell RNA sequencing data: challenges and opportunities (2016). Catalina Vallejos, Davide Risso, Antonio Scialdone, Sandrine Dudoit and John Marioni

Single-cell RNA-seq (scRNA-seq) datasets are typically normalized using methods that were developed for bulk RNA-seq data. Here, we apply to two real datasets the most widely used global-scaling normalization techniques: Reads Per Million (RPM), DESeq, and Trimmed Mean of M-values (TMM) (see manuscript for references and a brief description). The datasets were chosen to reflect the wide range of scRNA-seq data present in the literature:

- Dataset 1: mESCs from (Klein et al. 2015): 933 cells collected with the inDrop protocol (droplet-based technology) and sequenced at the average depth of 24,000 reads per cell.
- Dataset 2: mouse Embryonic Stem Cells (mESCs) from (Kolodziejczyk et al. 2015): 705 cells collected with the Fluidigm C1 (microfluidics-based) technology and sequenced at the average depth of 3.6 million reads per cell.
- Dataset 3: mouse visual cortex cells from (Tasic et al. 2016): 1,679 cells collected after FACS sorting into 96-well plates and sequenced at the average depth of 5.7 million reads per cell.
- Dataset 4: mouse cortex and hippocampus cells from (Zeisel et al. 2015): 3,005 cells collected with the Fluidigm C1 (microfluidics-based) technology and sequenced at the average depth of 14,000 reads per cell (with UMI).

The first two datasets are characterized by a homogeneous cell population (mESC), measured with two different technologies (Fluidigm vs inDrop). The third and fourth datasets are collected from a mixture of heterogeneous cell populations (multiple brain tissues, *in vivo*). The four datasets have different sequencing depths and span the most widely used cell capturing methods: droplet-based (Dataset 1), Fluidigm C1 microfluidics (Dataset 2 and 4), and manual plate-based methods (Dataset 3).

Dataset 1

Loading data and filtering out genes. We only selected the “Day 0” samples (933 cells).

```
esd0 <- read.csv("../datasets/Klein2015/GSM1599494_ES_d0_main.csv.bz2",
                 header=FALSE, stringsAsFactors=FALSE, row.names=1)
colnames(esd0) <- paste0("d0.", seq_len(ncol(esd0)))

counts<-esd0

save(list="counts", file="../datasets/Klein2015/klein.RData")

load("../datasets/Klein2015/klein.RData")
Cell.Colour<-"plum"
```

Normalization and gene filtering

We apply a filtering to remove lowly expressed genes with the function “filter.genes” (see below “Function Definitions” in the “Appendix” section): first, we remove the genes that are not detected in any cell, then we take the 50% of genes with the highest normalized average expression (in reads per million).

```
RawCounts<-counts
Gene.Ids<-row.names(RawCounts)
```

```

filter<-filter.genes(RawCounts)

Select<-filter$Select

RawCountsDropFilter <- RawCounts[Select,]

```

A number N=12020 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 25.21 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM. Note that the scaling factors computed by each method are normalised such that their average is equal to 1.

```

NormRpm <- rpm(RawCountsDropFilter)
NormDeseq <- deseq(RawCountsDropFilter)
NormTmm <- tmm(RawCountsDropFilter)

```

Number of genes used for DESeq normalization:

```

test<-apply(RawCountsDropFilter, 1, function(x) length(which(x==0)))
length(test[test==0])

## [1] 115

```

Comparing normalization strategies

Let us compare the size factors computed with the different methods.

```

#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsDropFilter, 2,
                   function(x) length(which(x==0))/nrow(RawCountsDropFilter))

min(frac.zeros)

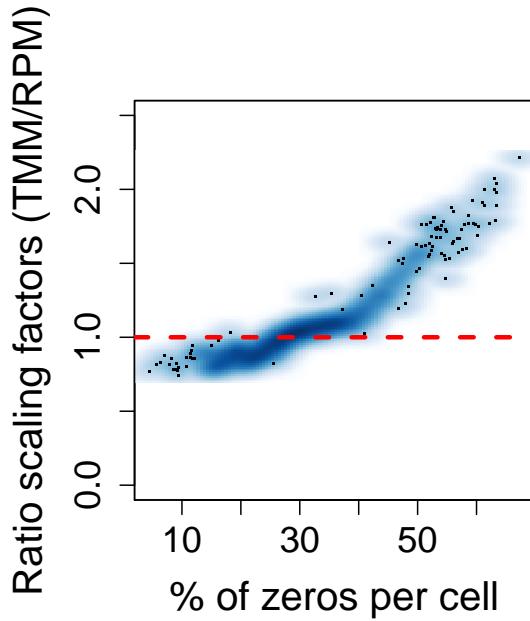
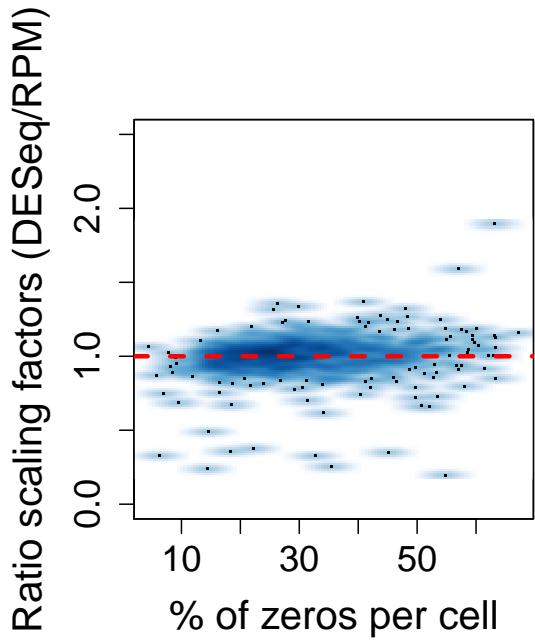
## [1] 0.04467554
max(frac.zeros)

## [1] 0.6722962

df<-data.frame(zeros = frac.zeros,
                x = NormRpm$s,
                y = NormDeseq$s,
                z = NormTmm$s)

plot.sf.ratio.frac.zeros.cata(df, ylim.max = 2.5)

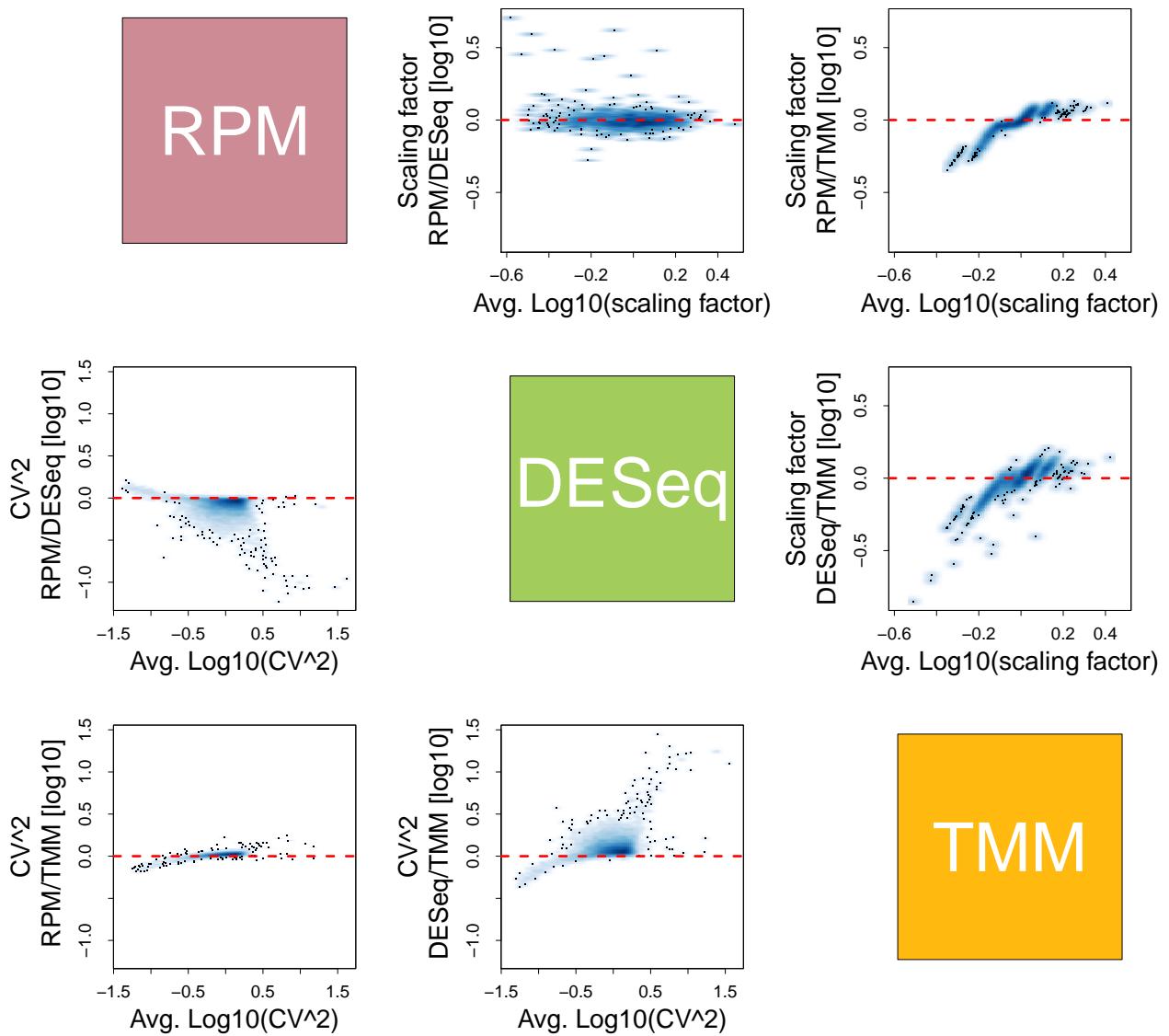
```



```

CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

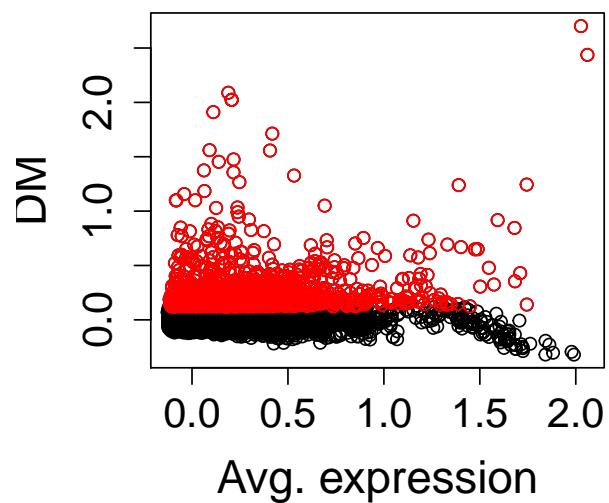
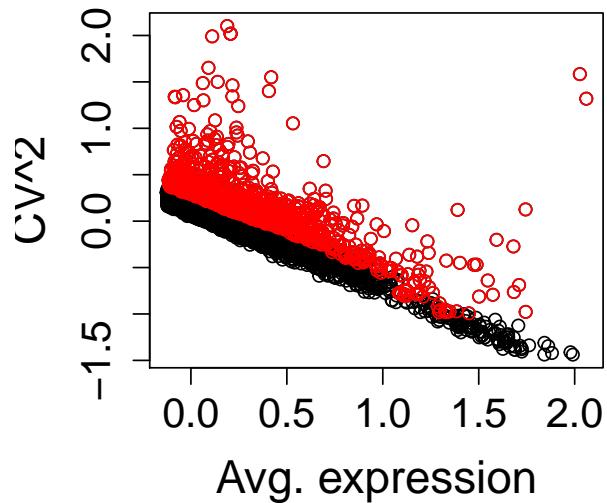
df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")
```



Highly variable genes detection

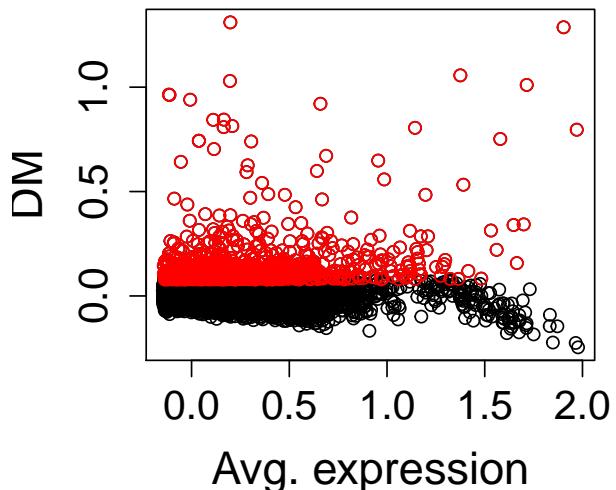
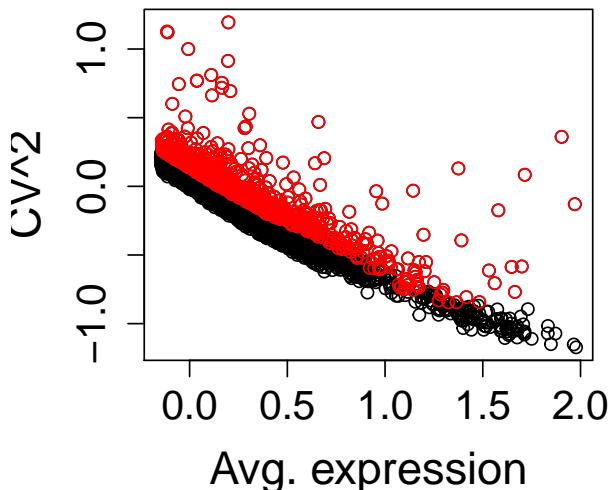
```
par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(f*length(Select))
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")
```

DESeq

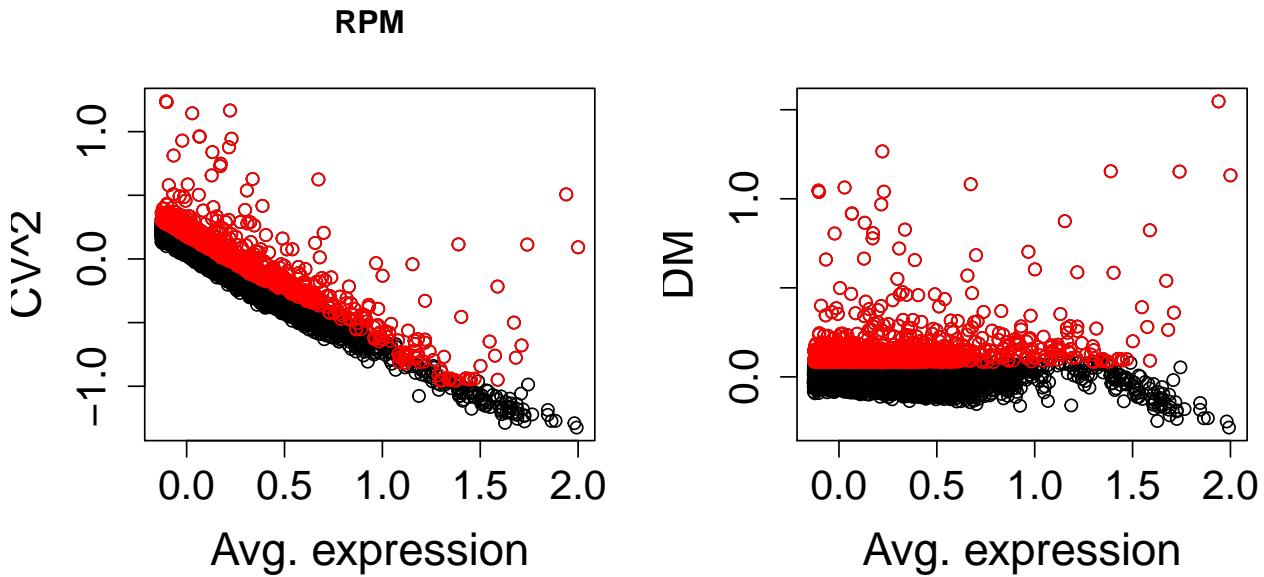


```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM

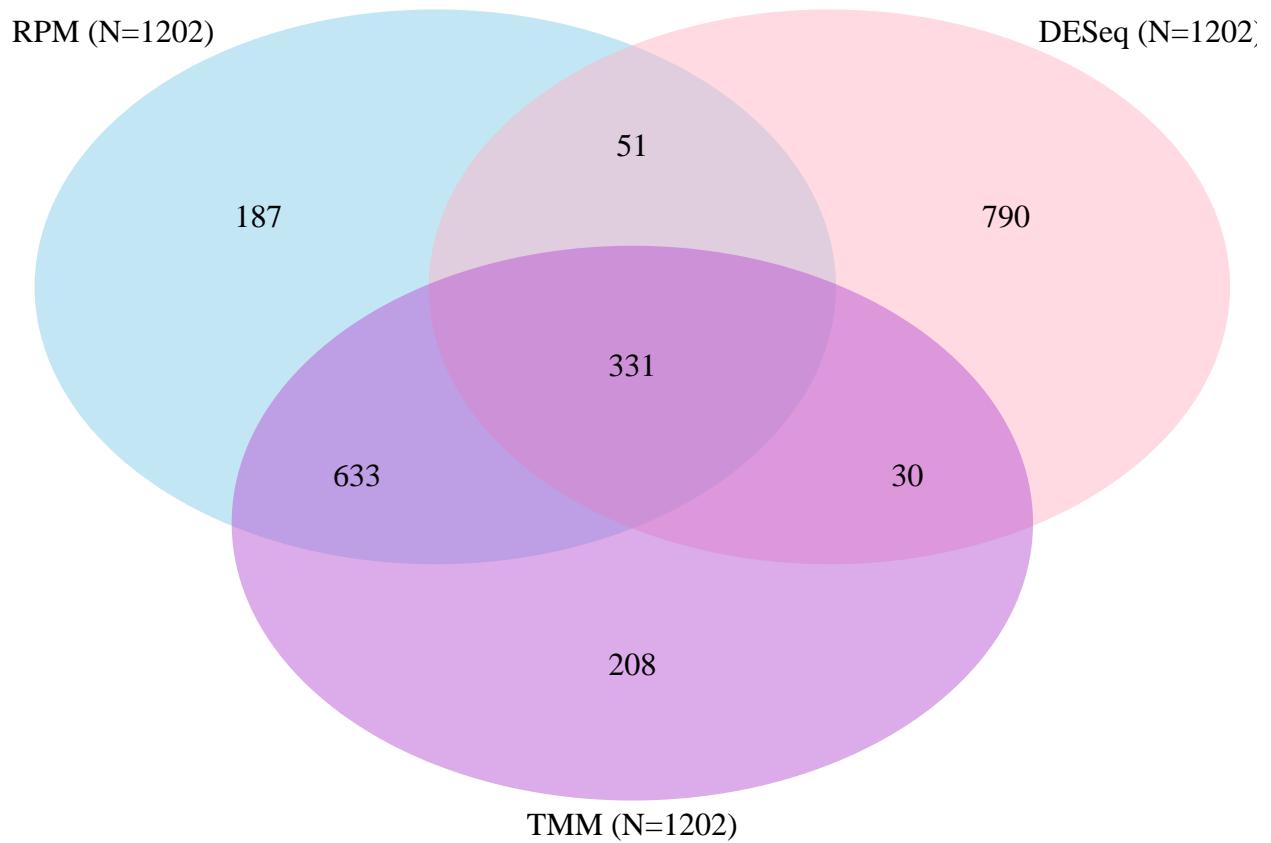


```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```



```
#define a list of highly variable genes
list.hvg<-list(rpm=hvg.rpm$genes.high.var,
                deseq=hvg.deseq$genes.high.var,
                tmm=hvg.tmm$genes.high.var)

grid.newpage()
draw.triple.venn(area1 = length(list.hvg$rpm),
                  area2 = length(list.hvg$deseq),
                  area3 = length(list.hvg$tmm),
                  n12 = length(intersect(list.hvg$rpm,list.hvg$deseq)),
                  n23 = length(intersect(list.hvg$deseq,list.hvg$tmm)),
                  n13 = length(intersect(list.hvg$rpm,list.hvg$tmm)),
                  n123=length(Intersect(list.hvg)),
                  category = c(paste0("RPM (N=",length(list.hvg$rpm), ")"),
                               paste0("DESeq (N=",length(list.hvg$deseq), ")"),
                               paste0("TMM (N=",length(list.hvg$tmm), ")")),
                  lty = "blank",
                  fill = c("skyblue", "pink1", "mediumorchid"))
```



```
## (polygon[GRID.polygon.11], polygon[GRID.polygon.12], polygon[GRID.polygon.13], polygon[GRID.polygon.14])
331 out of 1202 highly variable genes (28%) are shared across the three normalization methods.
```

Dataset 2

First, we select a subset of 79 cells to analyze (the first replicate of the “serum” condition).

```
all.counts<-fread("../datasets/counttable_es.csv", header=TRUE )

#select cell types and batch
cell.type<-"_lif_3"

Gene.Ids <- all.counts$gene
#Gene.IDs include spikes and several statistics from gene counting (unaligned reads, etc.)

Cell.Ids <- names(all.counts)[-1]

#take only endogenous genes
RawCounts <- subset(x=all.counts, subset=grep("ENSMUSG", Gene.Ids),
                      select = Cell.Ids[grep(pattern=cell.type, x=Cell.Ids)]) 

Cell.Ids<-colnames(RawCounts)

Cell.Colour<-"darkorange"
```

```

Gene.Ids<-Gene.Ids[grep("ENSMUSG", Gene.Ids)]
#Gene.IDs include spikes and several statistics
#from gene counting (unaligned reads, etc.): remove them

RawCounts<-as.matrix(RawCounts)

row.names(RawCounts)<-Gene.Ids

```

Normalization and gene filtering

Here we remove the lowly expressed genes (see “Dataset 1” for a description of the filtering):

```

filter<-filter.genes(RawCounts)
Select<-filter$Select
RawCountsOlaFilter <- RawCounts[Select,]
Gene.Ids <- Gene.Ids[Select]

```

A number N=12714 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 2.26 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM.

```

NormRpm <- rpm(RawCountsOlaFilter)
NormDeseq <- deseq(RawCountsOlaFilter)
NormTmm <- tmm(RawCountsOlaFilter)

```

Number of genes used for DESeq normalization:

```

test<-apply(RawCountsOlaFilter, 1, function(x) length(which(x==0)))
length(test[test==0])

## [1] 3387

```

Comparing normalization strategies

Let us compare cell-specific size factors and gene-specific coefficient of variation computed with the different methods.

```

#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsOlaFilter, 2,
                   function(x) length(which(x==0))/nrow(RawCountsOlaFilter))
min(frac.zeros)

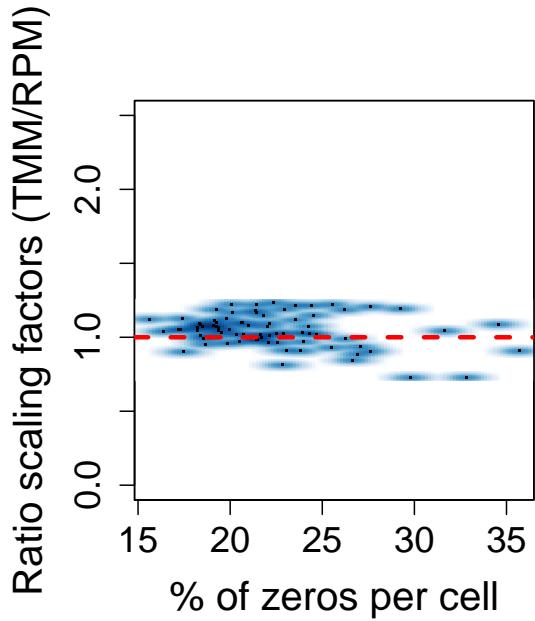
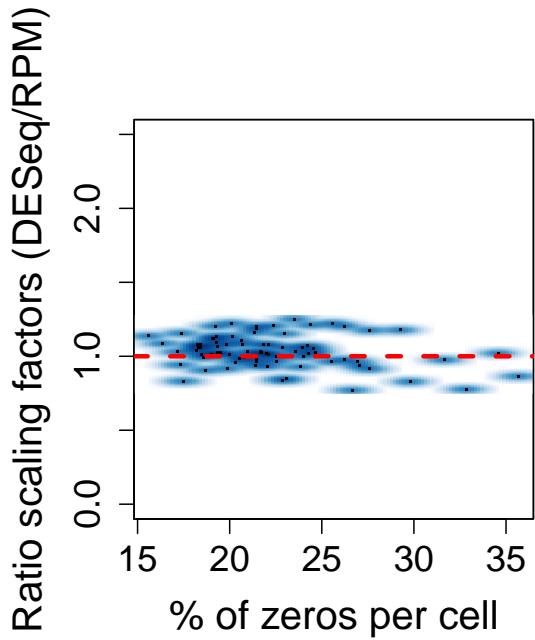
## [1] 0.1561271
max(frac.zeros)

## [1] 0.3568507

df<-data.frame(zeros = frac.zeros,
                 x = NormRpm$s,
                 y = NormDeseq$s,
                 z = NormTmm$s)

plot.sf.ratio.fraction.zeros.cata(df, ylim.max = 2.5)

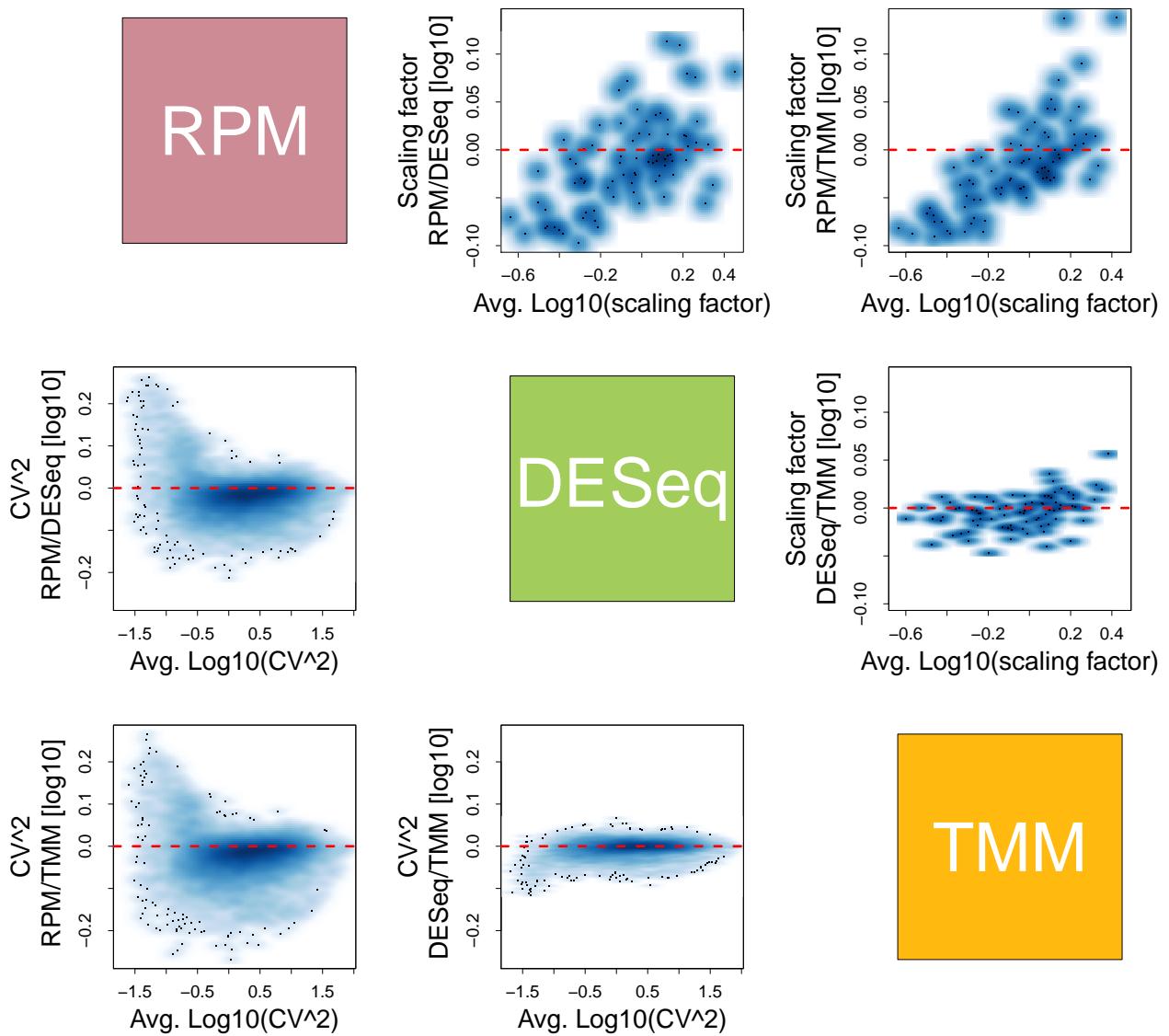
```



```

CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")
```

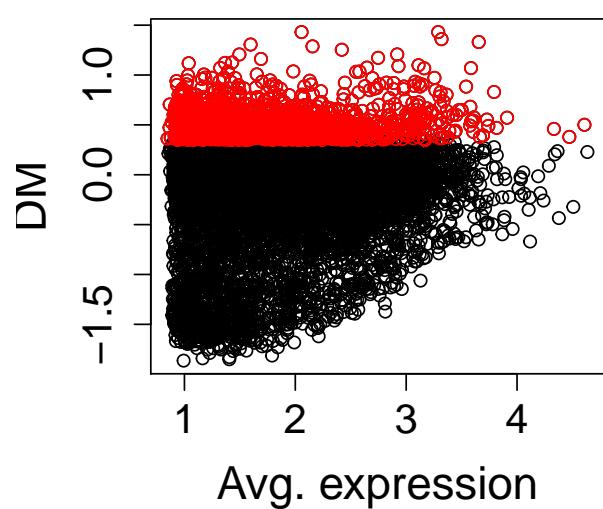
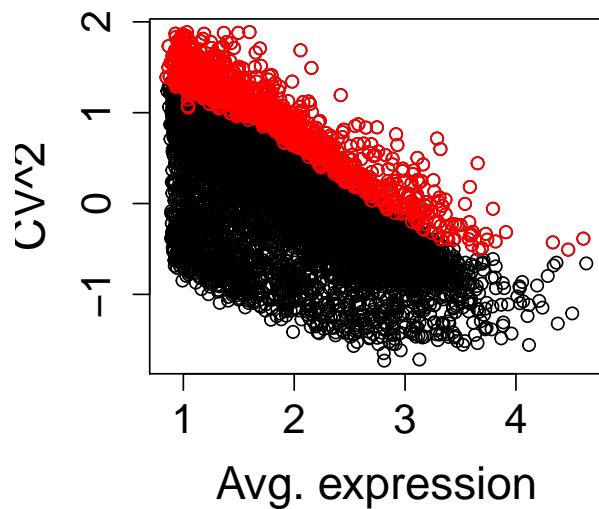


Highly variable genes detection

We use the distance-to-median (DM) method to select the most highly variable genes.

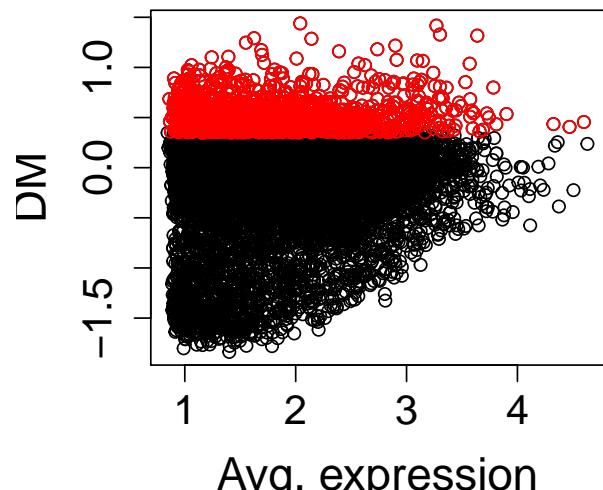
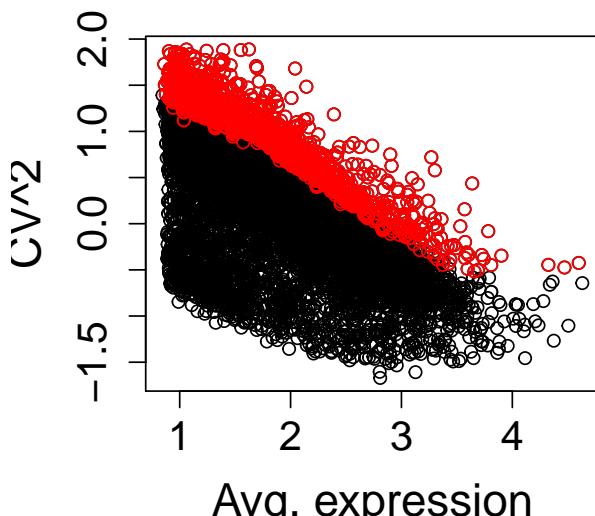
```
par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(length(Select)*f)#1e3
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")
```

DESeq

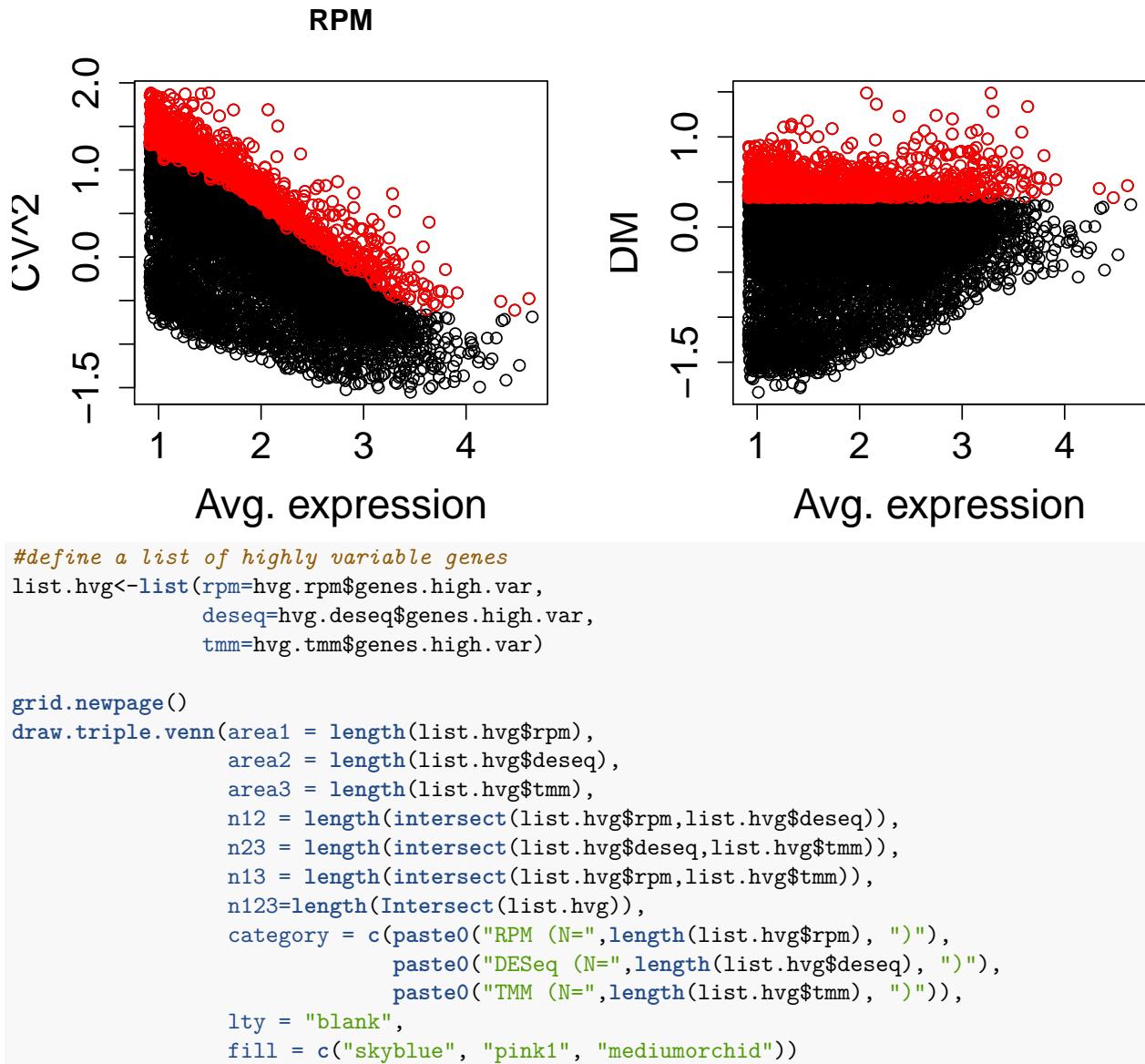


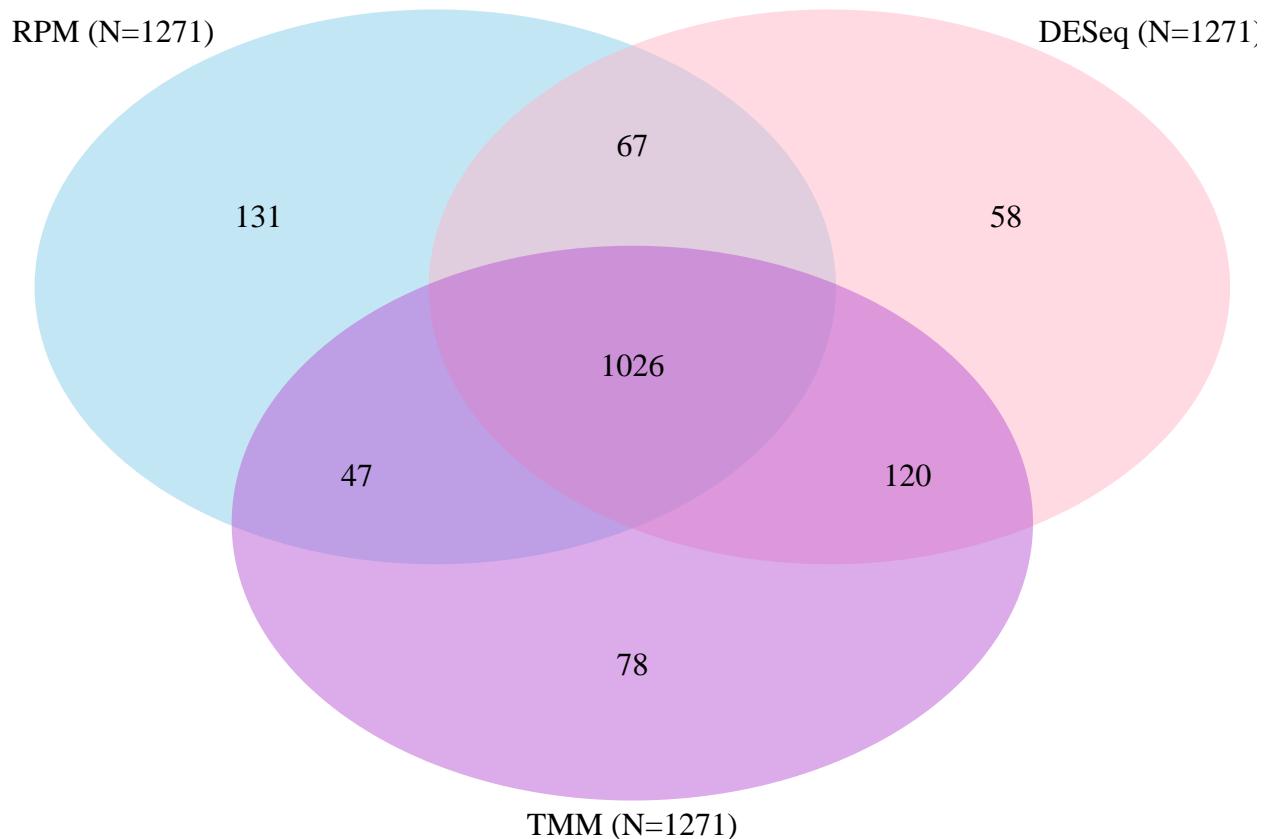
```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM



```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```





```
## (polygon[GRID.polygon.27], polygon[GRID.polygon.28], polygon[GRID.polygon.29], polygon[GRID.polygon.30])
1026 out of 1271 highly variable genes (81%) are shared across the three normalization methods.
```

Dataset 3

Loading data and filtering out genes. We only selected the 190 cells from the Rbp4 CRE line to have a more homogeneous cell population.

```
library(scRNASeq)
data(alen)
Cell.Colour<-"aquamarine3"
```

Normalization and gene filtering

First, we remove the lowly expressed genes (see “Dataset 1” for a description of the filtering):

```
RawCounts<-assay(alen[, colData(alen)$driver_1_s=="Rbp4-Cre_KL100"])
RawCounts<-RawCounts[grep("ERCC", rownames(RawCounts), invert = TRUE),]

filter<-filter.genes(RawCounts)

Select<-filter$Select

RawCountsAllenFilter <- RawCounts[Select,]
```

A number N=8545 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 8.14 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM.

```
NormRpm <- rpm(RawCountsAllenFilter)
NormDeseq <- deseq(RawCountsAllenFilter)
NormTmm <- tmm(RawCountsAllenFilter)
```

Number of genes used for DESeq normalization:

```
test<-apply(RawCountsAllenFilter, 1, function(x) length(which(x==0)))
length(test[test==0])
## [1] 798
```

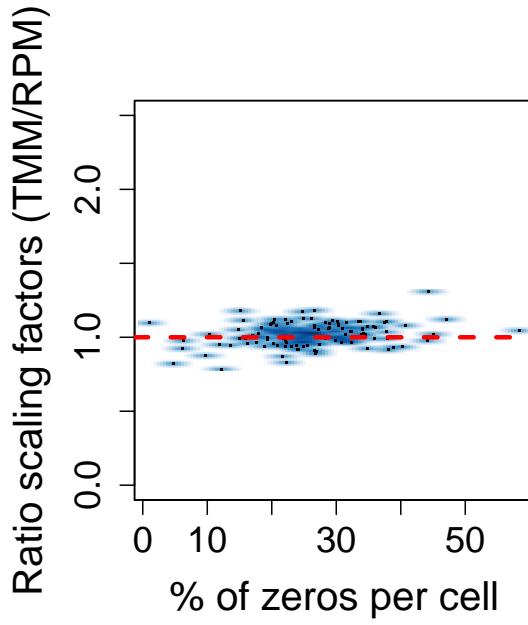
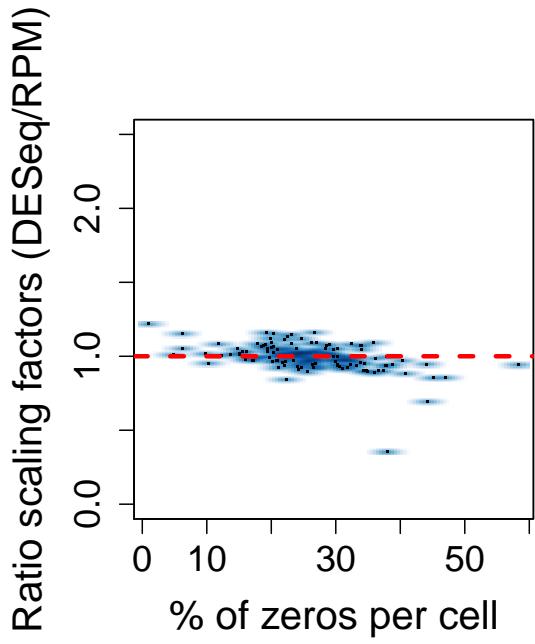
Comparing normalization strategies

Let us compare the size factors computed with the different methods.

```
#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsAllenFilter, 2,
                    function(x) length(which(x==0))/nrow(RawCountsAllenFilter))
min(frac.zeros)
## [1] 0.01018139
max(frac.zeros)
## [1] 0.5834991

df<-data.frame(zeros = frac.zeros,
                 x = NormRpm$s,
                 y = NormDeseq$s,
                 z = NormTmm$s)

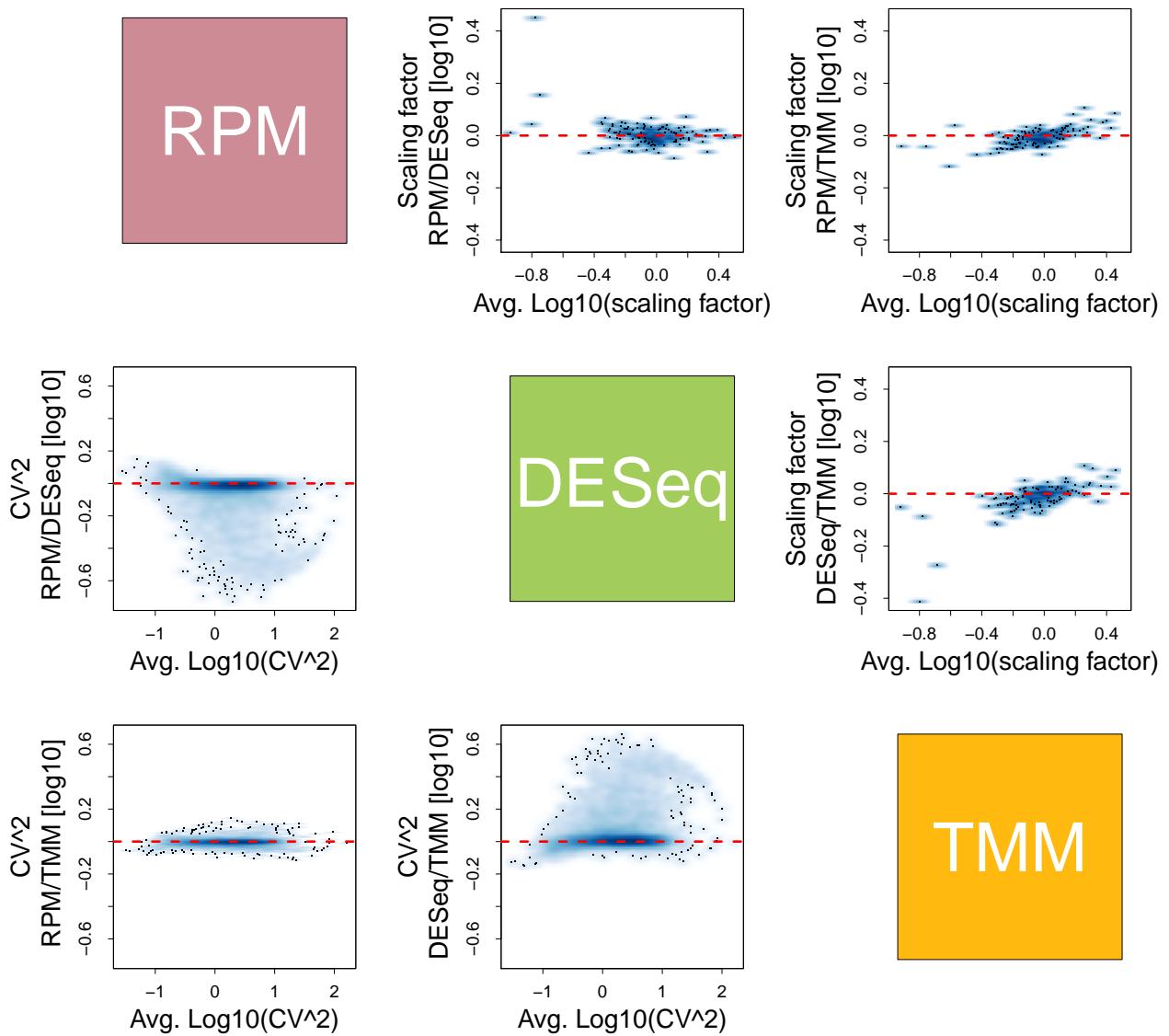
plot.sf.ratio.frac.zeros.cata(df, ylim.max = 2.5)
```



```

CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")
```



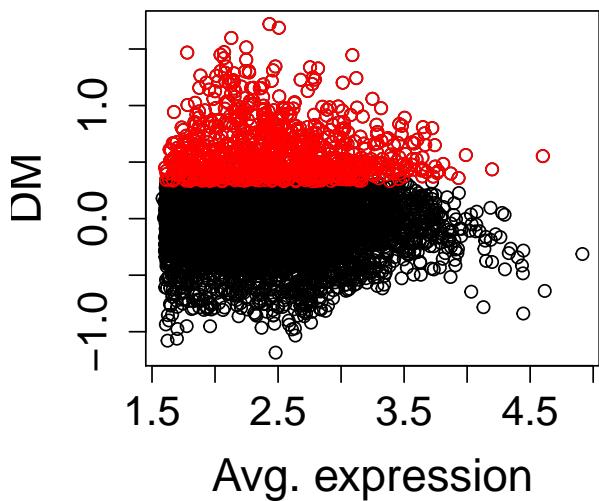
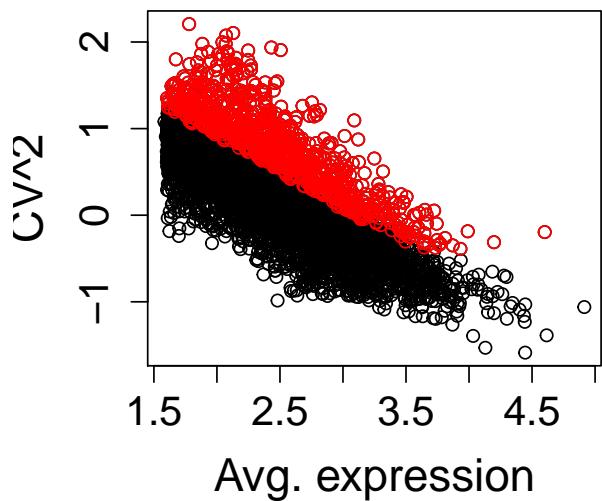
Highly variable genes detection

```

par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(f*length(Select))
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")

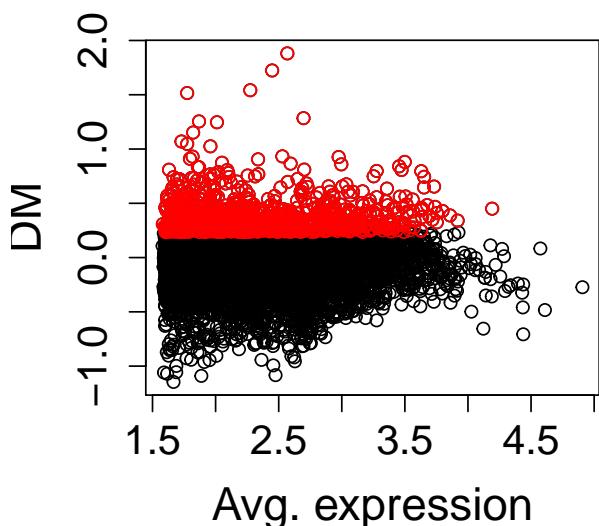
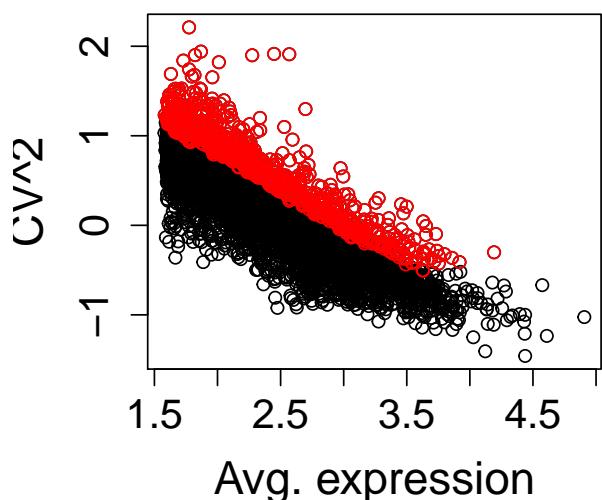
```

DESeq

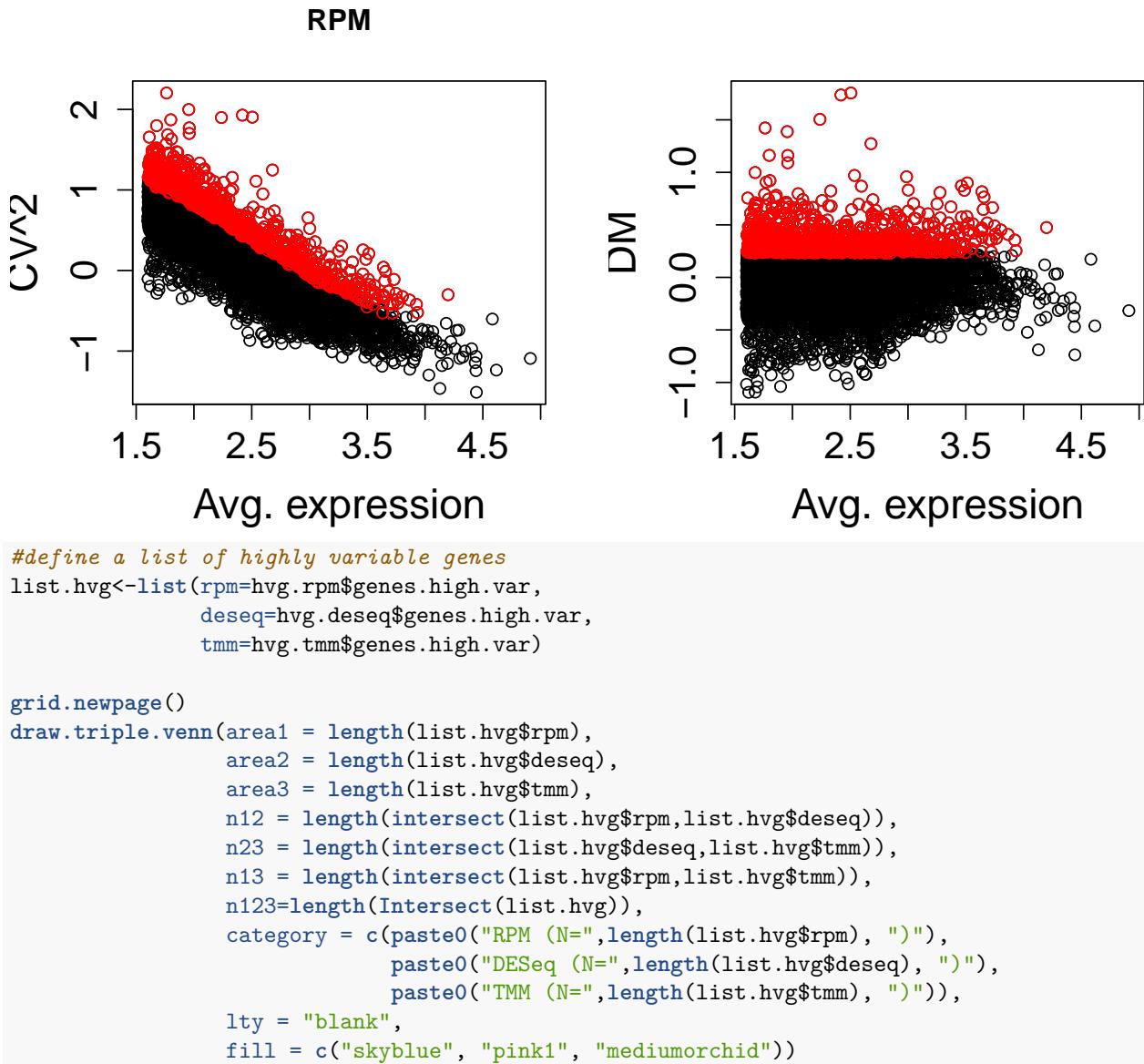


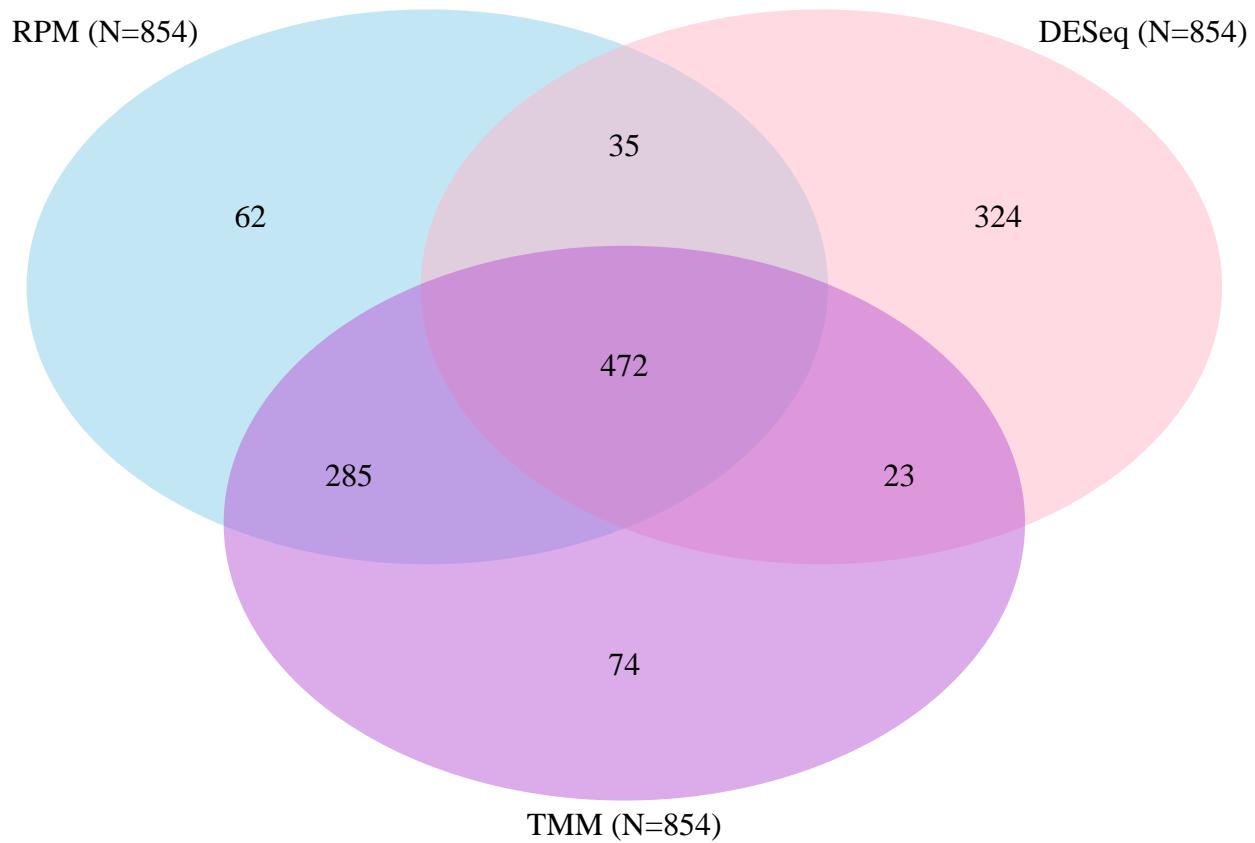
```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM



```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```





```
## (polygon[GRID.polygon.43], polygon[GRID.polygon.44], polygon[GRID.polygon.45], polygon[GRID.polygon.46])
472 out of 854 highly variable genes (55%) are shared across the three normalization methods.
```

Dataset 4

Loading data and filtering out genes. In order to consider homogeneous groups of cells, we select three of the subpopulations identified in the original publication (“astrocytes_ependymal”, “oligodendrocytes” and “microglia”) and analyse them separately.

```
zei <- read.table("../datasets/Zeisel2015/expression_mRNA_17-Aug-2014.txt",
                  header=FALSE, stringsAsFactors=FALSE, row.names=1, sep='\t', skip=11) [,-1]
colnames(zei) <- paste0("zei_", seq_len(ncol(zei)))

tmp <- read.table("../datasets/Zeisel2015/expression_mRNA_17-Aug-2014.txt",
                  nrows=3, skip=7, sep='\t', stringsAsFactors = FALSE)
tmp <- as.matrix(tmp[,-(1:2)])
idx.astro <- which(tmp[2]== "astrocytes_ependymal")
idx.micro <- which(tmp[2]== "microglia")
idx.oligo <- which(tmp[2]== "oligodendrocytes")

save(zei, idx.astro,
      idx.micro, idx.oligo,
      file="../datasets/Zeisel2015/zeisel.RData")

load("../datasets/Zeisel2015/zeisel.RData")
Cell.Colour<-"lightblue"
```

Subpopulation 1: astrocytes-ependymal

Normalization and gene filtering

First, we remove the lowly expressed genes (see “Dataset 1” for a description of the filtering):

```
RawCounts<-zei[,idx.astro]

filter<-filter.genes(RawCounts)
Select<-filter$Select

RawCountsZeiFilter <- RawCounts[Select,]
```

A number N=7971 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 17.57 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM.

```
NormRpm <- rpm(RawCountsZeiFilter)
NormDeseq <- deseq(RawCountsZeiFilter)
NormTmm <- tmm(RawCountsZeiFilter)
```

Number of genes used for DESeq normalization:

```
test<-apply(RawCountsZeiFilter, 1, function(x) length(which(x==0)))
length(test[test==0])

## [1] 3
```

Comparing normalization strategies

Let us compare the size factors computed with the different methods.

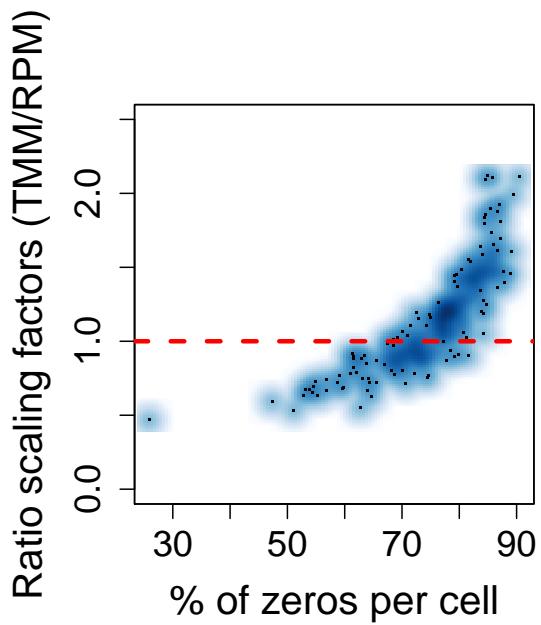
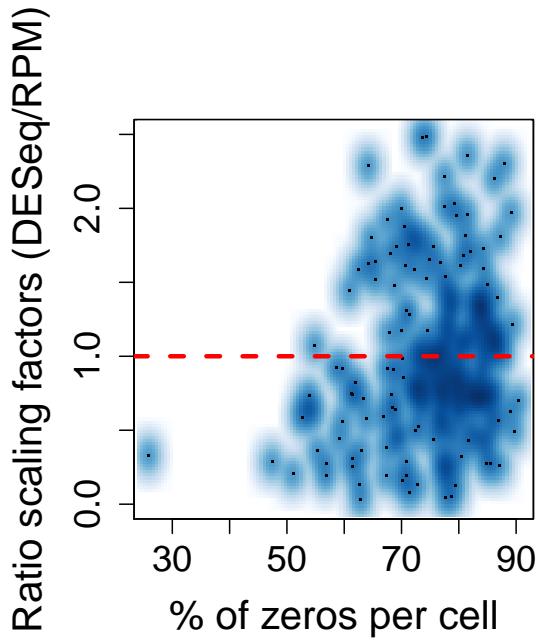
```
#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsZeiFilter, 2,
                   function(x) length(which(x==0))/nrow(RawCountsZeiFilter))
min(frac.zeros)

## [1] 0.2590641
max(frac.zeros)

## [1] 0.9044035

df<-data.frame(zeros = frac.zeros,
                 x = NormRpm$s,
                 y = NormDeseq$s,
                 z = NormTmm$s)

plot.sf.ratio.fraction(df, ylim.max = 2.5)
```

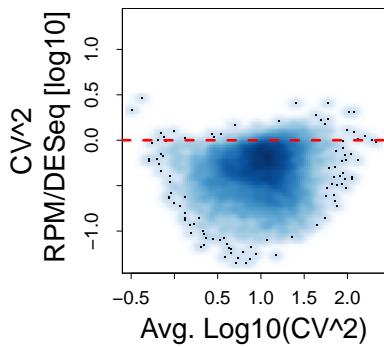
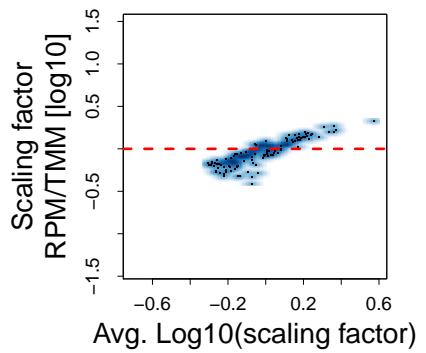
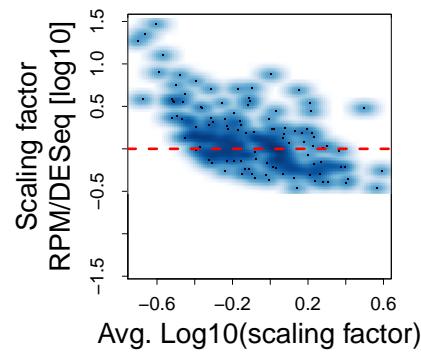


```

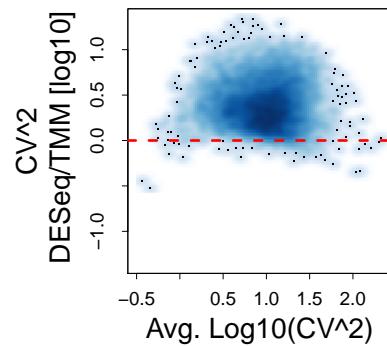
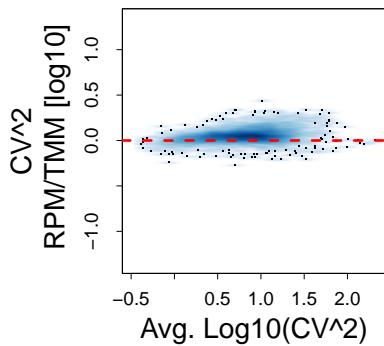
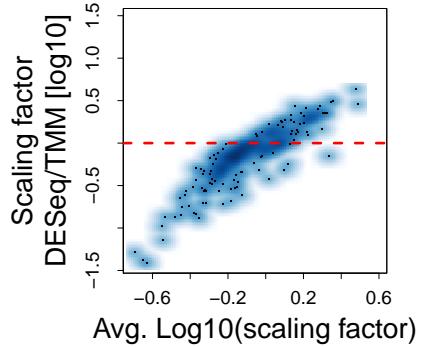
CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")
```

RPM



DESeq

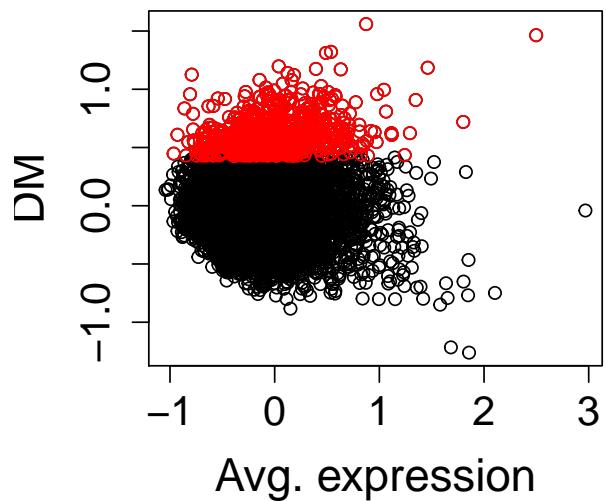
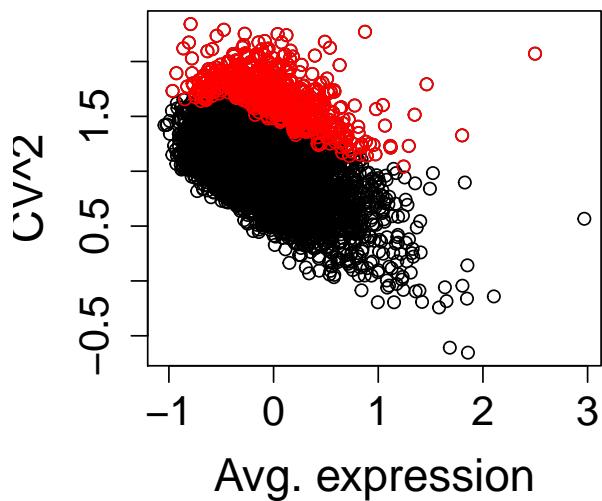


TMM

Highly variable genes detection

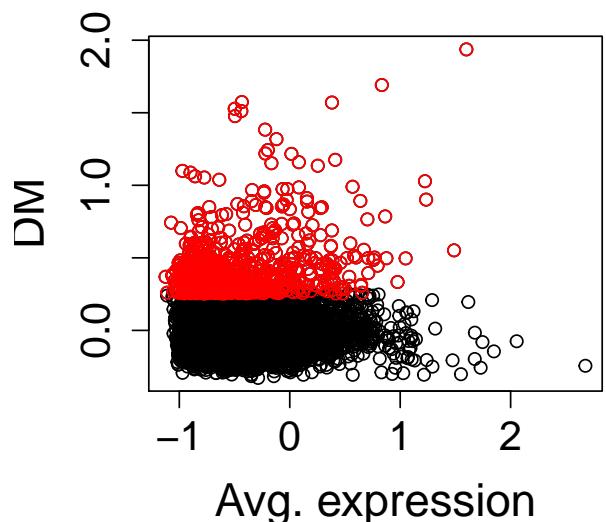
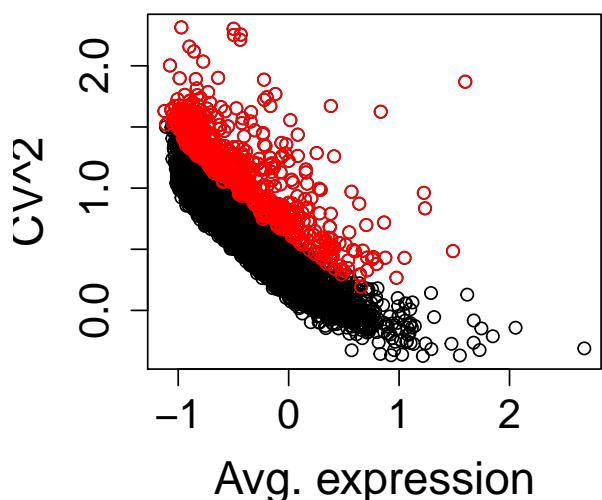
```
par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(f*length(Select))
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")
```

DESeq

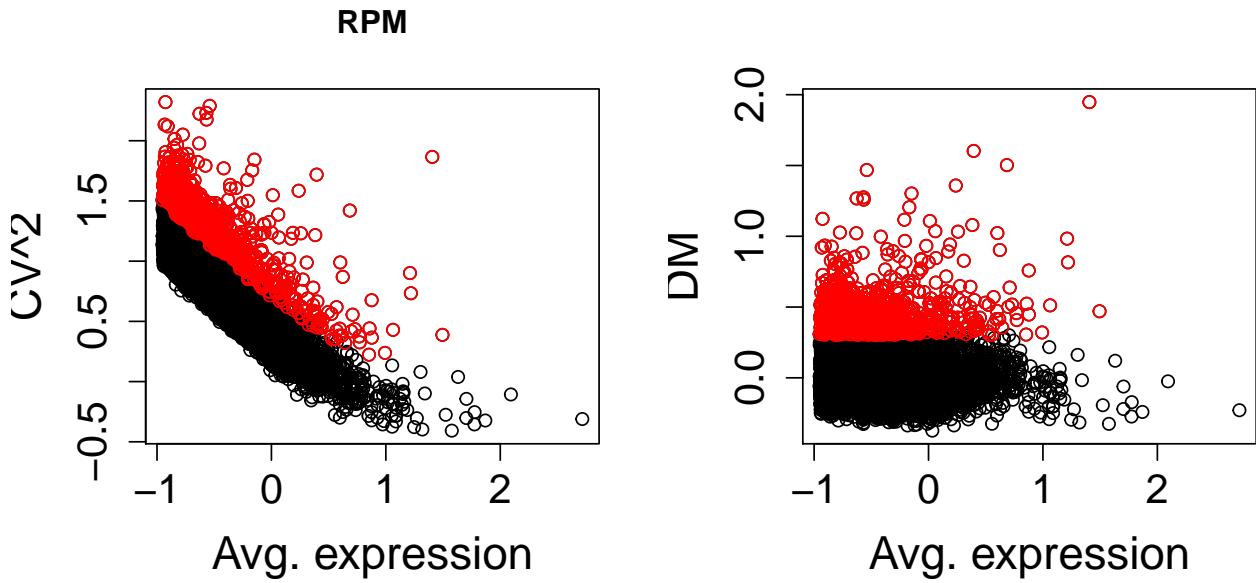


```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM

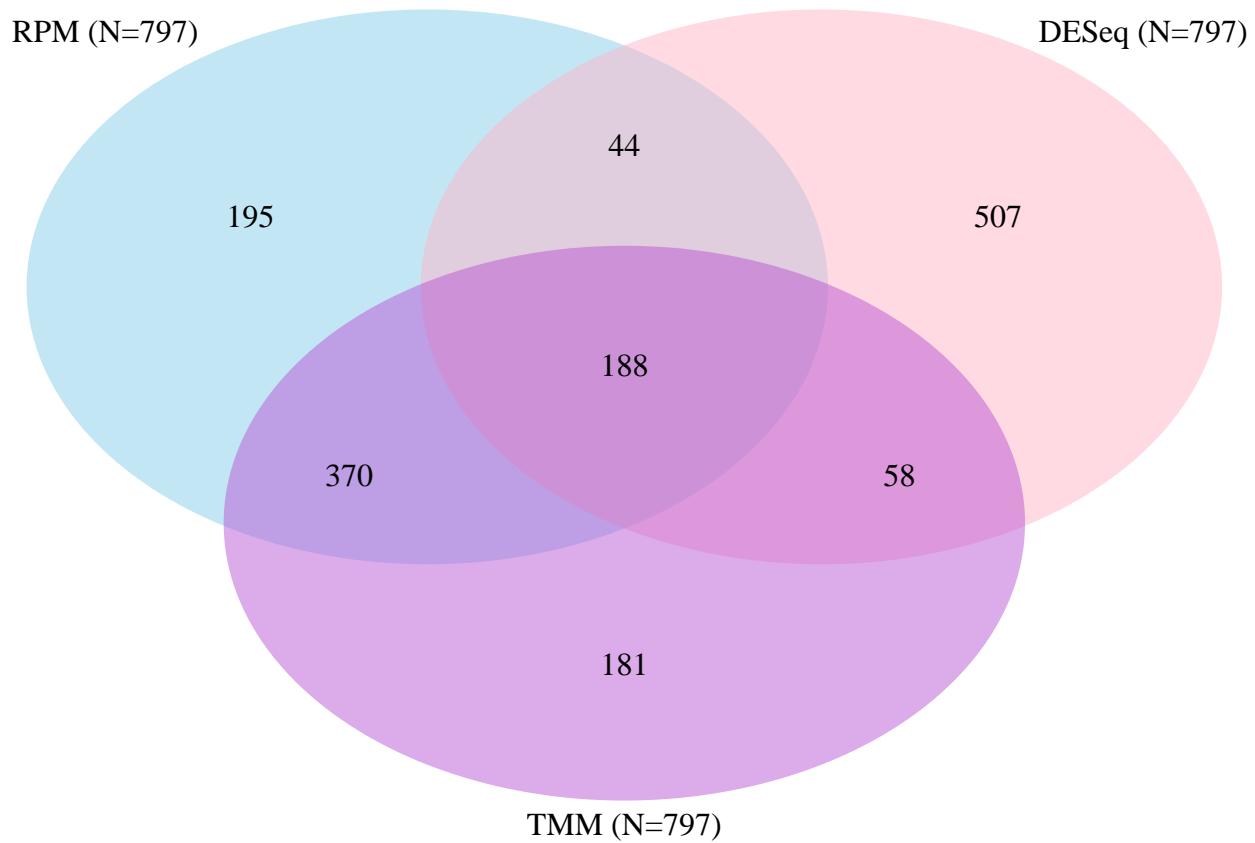


```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```



```
#define a list of highly variable genes
list.hvg<-list(rpm=hvg.rpm$genes.high.var,
                deseq=hvg.deseq$genes.high.var,
                tmm=hvg.tmm$genes.high.var)

grid.newpage()
draw.triple.venn(area1 = length(list.hvg$rpm),
                  area2 = length(list.hvg$deseq),
                  area3 = length(list.hvg$tmm),
                  n12 = length(intersect(list.hvg$rpm,list.hvg$deseq)),
                  n23 = length(intersect(list.hvg$deseq,list.hvg$tmm)),
                  n13 = length(intersect(list.hvg$rpm,list.hvg$tmm)),
                  n123=length(Intersect(list.hvg)),
                  category = c(paste0("RPM (N=",length(list.hvg$rpm), ")"),
                               paste0("DESeq (N=",length(list.hvg$deseq), ")"),
                               paste0("TMM (N=",length(list.hvg$tmm), ")")),
                  lty = "blank",
                  fill = c("skyblue", "pink1", "mediumorchid"))
```



```
## (polygon[GRID.polygon.59], polygon[GRID.polygon.60], polygon[GRID.polygon.61], polygon[GRID.polygon.62])
188 out of 797 highly variable genes (24%) are shared across the three normalization methods.
```

Subpopulation 2: oligodendrocytes

Normalization and gene filtering

First, we remove the lowly expressed genes (see “Dataset 1” for a description of the filtering):

```
RawCounts<-zei[,idx.oligo]

filter<-filter.genes(RawCounts)
Select<-filter$Select

RawCountsZeiFilter <- RawCounts[Select,]
```

A number N=8973 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 9.03 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM.

```
NormRpm <- rpm(RawCountsZeiFilter)
NormDeseq <- deseq(RawCountsZeiFilter)
NormTmm <- tmm(RawCountsZeiFilter)
```

Number of genes used for DESeq normalization:

```

test<-apply(RawCountsZeiFilter, 1, function(x) length(which(x==0)))
length(test[test==0])

## [1] 7

```

Comparing normalization strategies

Let us compare the size factors computed with the different methods.

```

#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsZeiFilter, 2,
                    function(x) length(which(x==0))/nrow(RawCountsZeiFilter))
min(frac.zeros)

## [1] 0.3913964

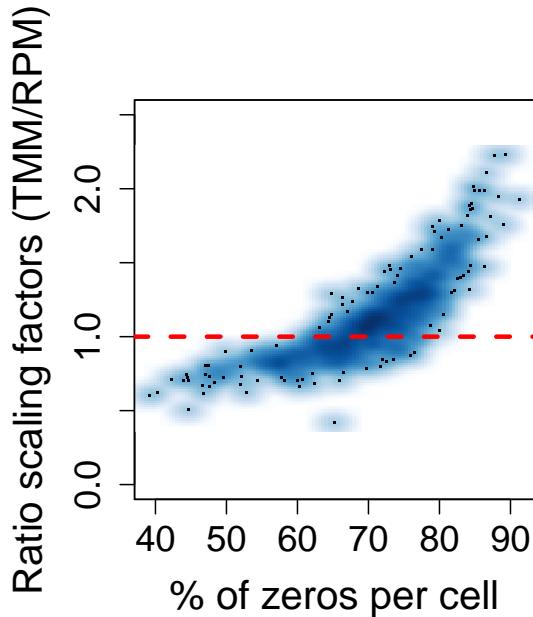
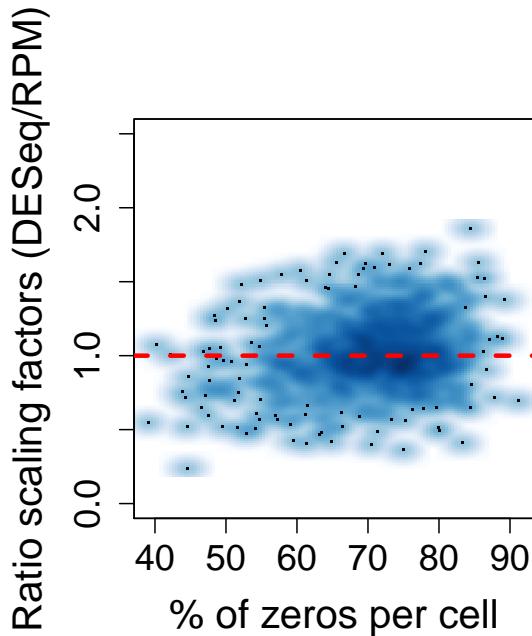
max(frac.zeros)

## [1] 0.912181

df<-data.frame(zeros = frac.zeros,
                 x = NormRpm$s,
                 y = NormDeseq$s,
                 z = NormTmm$s)

plot.sf.ratio.frac.zeros.cata(df, ylim.max = 2.5)

```

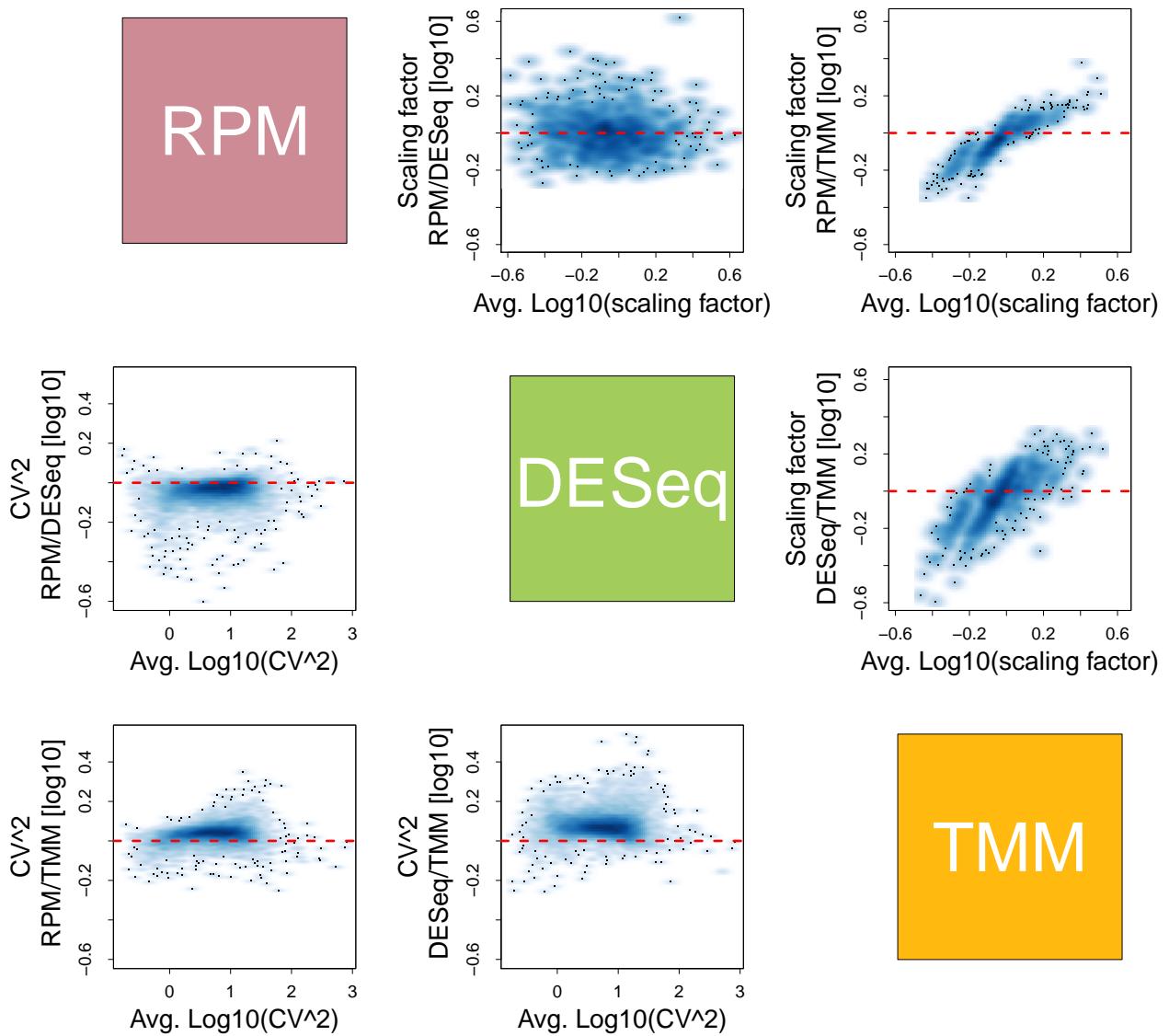


```

CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")

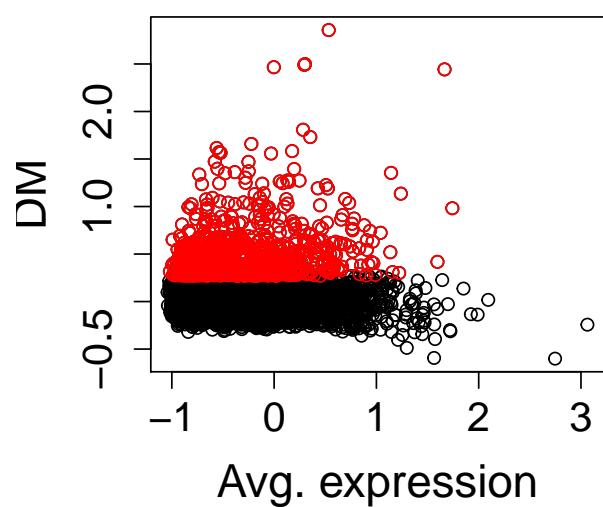
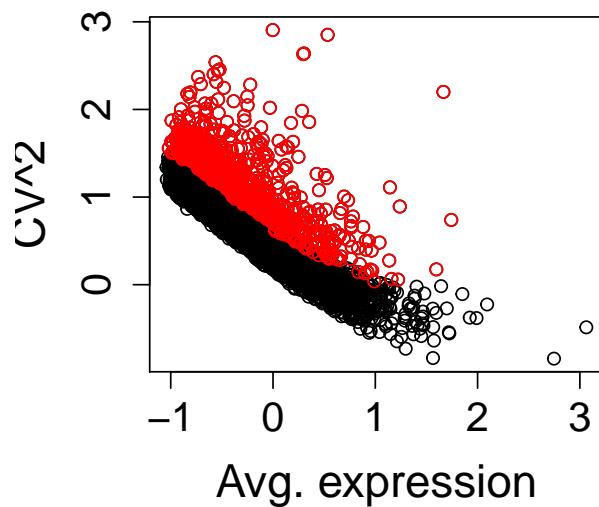
```



Highly variable genes detection

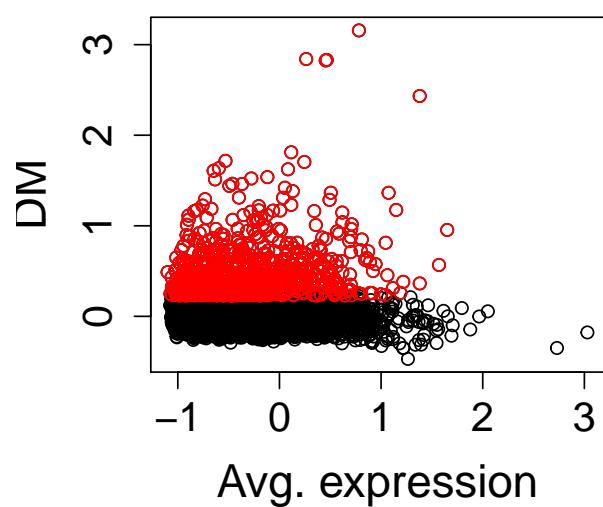
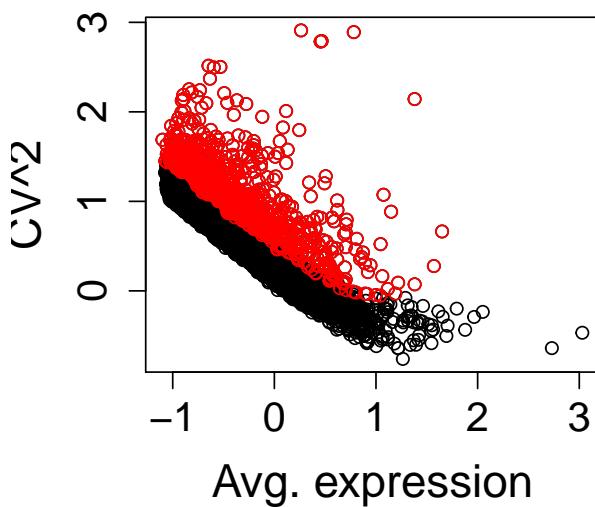
```
par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(f*length(Select))
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")
```

DESeq

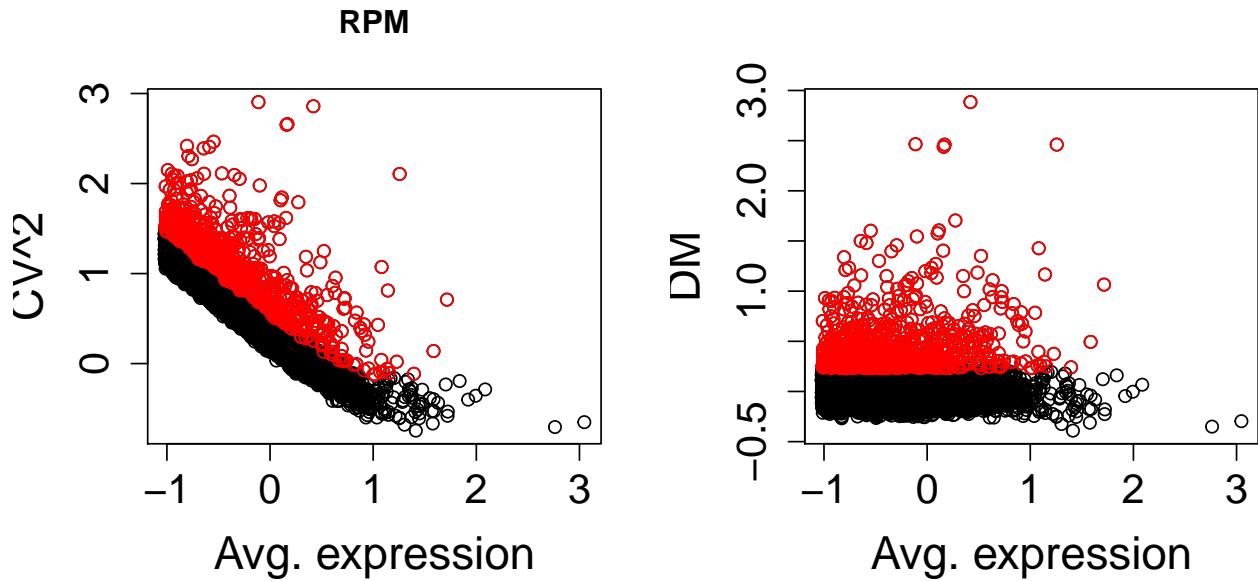


```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM

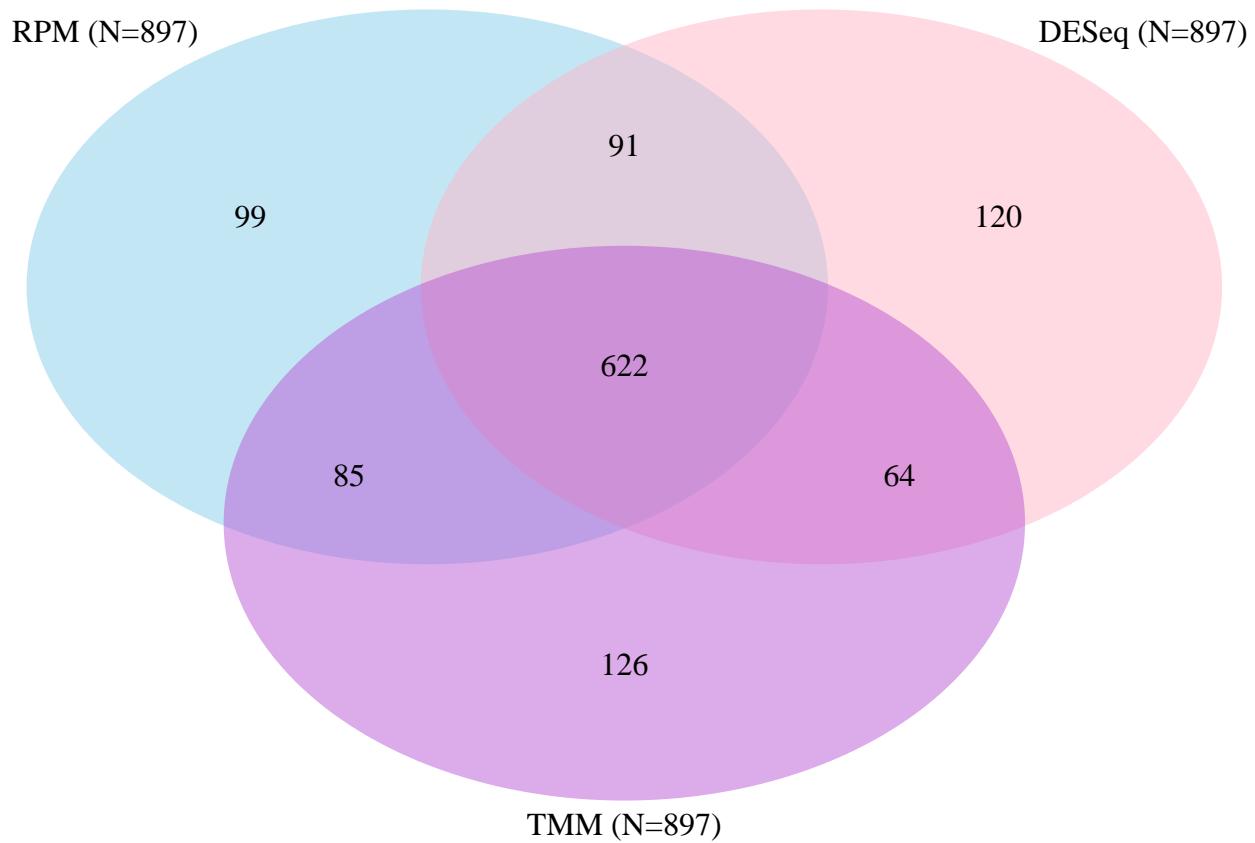


```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```



```
#define a list of highly variable genes
list.hvg<-list(rpm=hvg.rpm$genes.high.var,
                deseq=hvg.deseq$genes.high.var,
                tmm=hvg.tmm$genes.high.var)

grid.newpage()
draw.triple.venn(area1 = length(list.hvg$rpm),
                  area2 = length(list.hvg$deseq),
                  area3 = length(list.hvg$tmm),
                  n12 = length(intersect(list.hvg$rpm,list.hvg$deseq)),
                  n23 = length(intersect(list.hvg$deseq,list.hvg$tmm)),
                  n13 = length(intersect(list.hvg$rpm,list.hvg$tmm)),
                  n123=length(Intersect(list.hvg)),
                  category = c(paste0("RPM (N=",length(list.hvg$rpm), ")"),
                              paste0("DESeq (N=",length(list.hvg$deseq), ")"),
                              paste0("TMM (N=",length(list.hvg$tmm), ")")),
                  lty = "blank",
                  fill = c("skyblue", "pink1", "mediumorchid"))
```



```
## (polygon[GRID.polygon.75], polygon[GRID.polygon.76], polygon[GRID.polygon.77], polygon[GRID.polygon.78])
622 out of 897 highly variable genes (69%) are shared across the three normalization methods.
```

Subpopulation 3: microglia

Normalization and gene filtering

First, we remove the lowly expressed genes (see “Dataset 1” for a description of the filtering):

```
RawCounts<-zei[,idx.micro]

filter<-filter.genes(RawCounts)
Select<-filter$Select

RawCountsZeiFilter <- RawCounts[Select,]
```

A number N=7073 of genes was selected for downstream analysis. The minimum average expression among the selected genes is 21.42 reads per million.

We normalize the data by using DESeq size factors, TMM and RPM.

```
NormRpm <- rpm(RawCountsZeiFilter)
NormDeseq <- deseq(RawCountsZeiFilter)
NormTmm <- tmm(RawCountsZeiFilter)
```

Number of genes used for DESeq normalization:

```

test<-apply(RawCountsZeiFilter, 1, function(x) length(which(x==0)))
length(test[test==0])

## [1] 11

```

Comparing normalization strategies

Let us compare the size factors computed with the different methods.

```

#Count the fractions of zeros in each cell
frac.zeros<-apply(RawCountsZeiFilter, 2,
                    function(x) length(which(x==0))/nrow(RawCountsZeiFilter))
min(frac.zeros)

## [1] 0.5214195

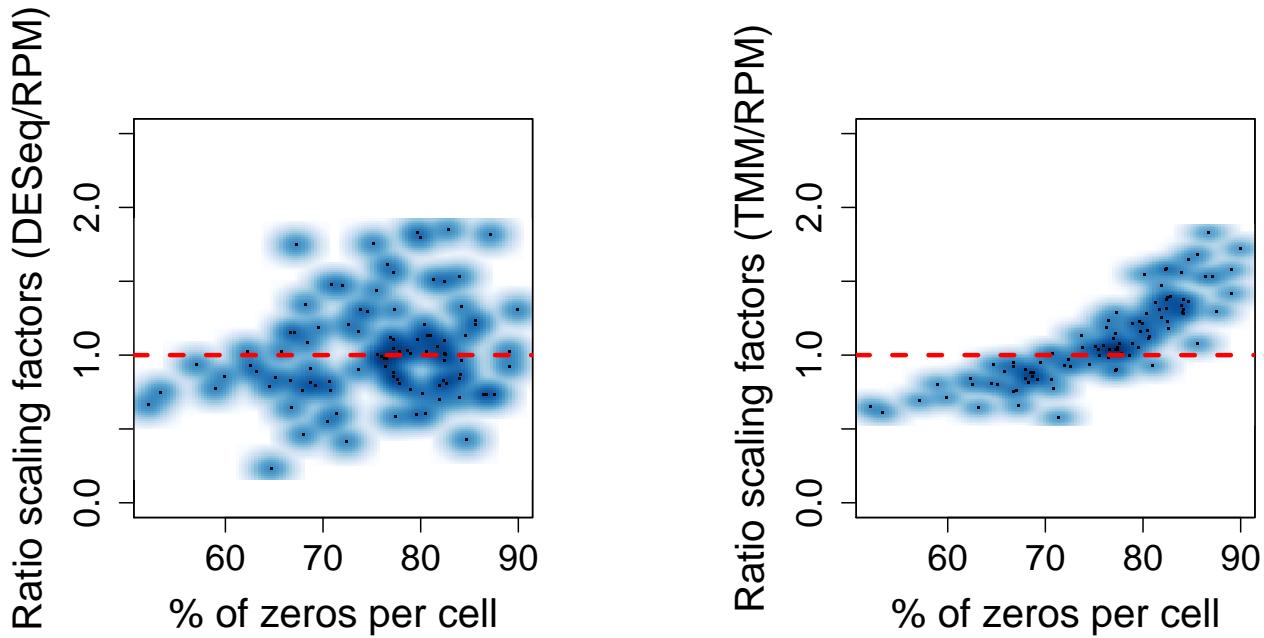
max(frac.zeros)

## [1] 0.8994769

df<-data.frame(zeros = frac.zeros,
                x = NormRpm$s,
                y = NormDeseq$s,
                z = NormTmm$s)

plot.sf.ratio.frac.zeros.cata(df, ylim.max = 2.5)

```

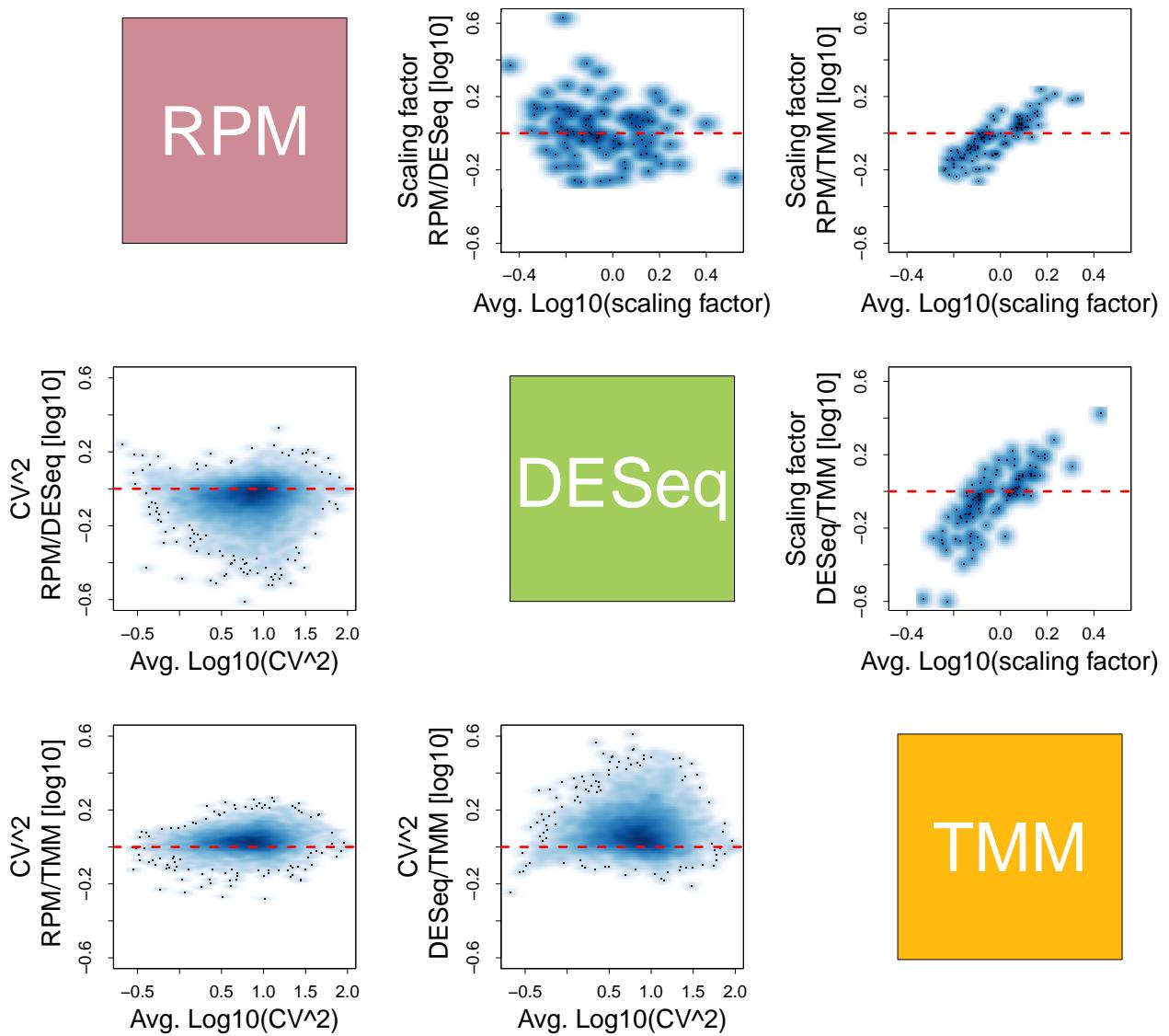


```

CVrpm<-apply(NormRpm$counts, 1, function(x) sd(x)/mean(x))
CVdeseq<-apply(NormDeseq$counts, 1, function(x) sd(x)/mean(x))
CVtmm<-apply(NormTmm$counts, 1, function(x) sd(x)/mean(x))

df1<-data.frame(x=NormRpm$s, y = NormDeseq$s, z = NormTmm$s)
df2<-data.frame(x=CVrpm^2, y = CVdeseq^2, z = CVtmm^2)
pairs.comparison.diff(df1, df2, title = "")

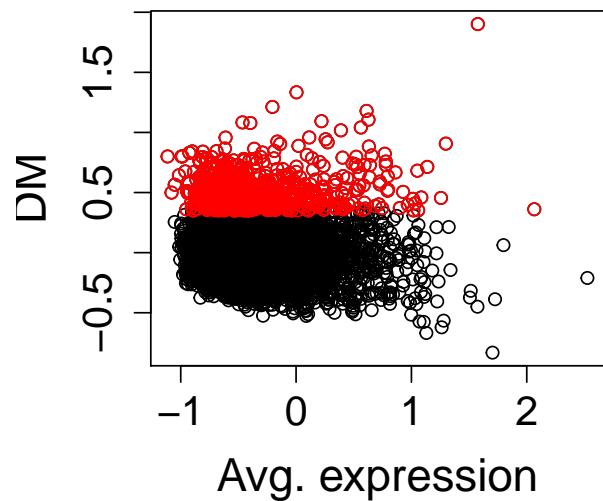
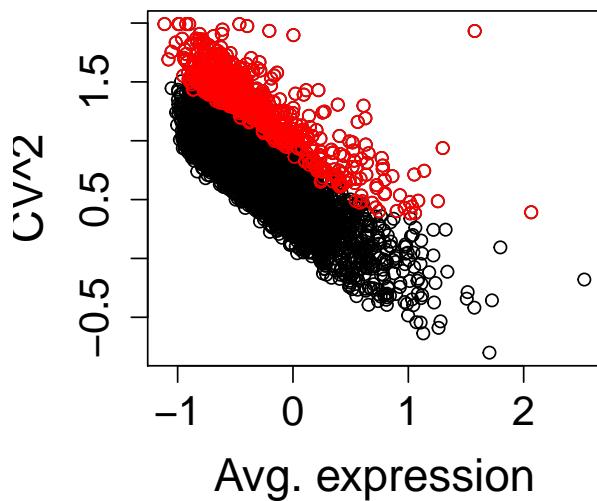
```



Highly variable genes detection

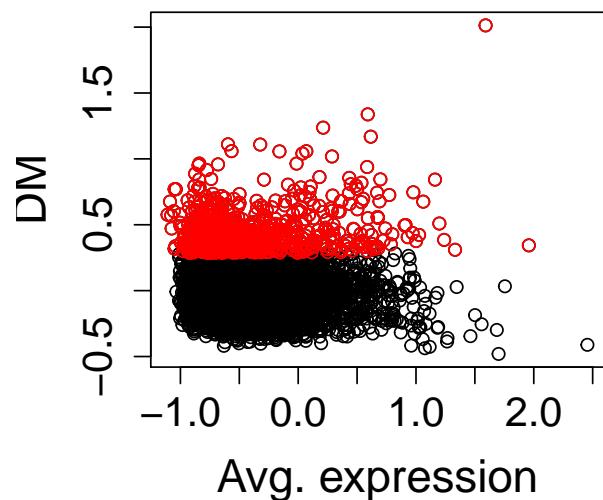
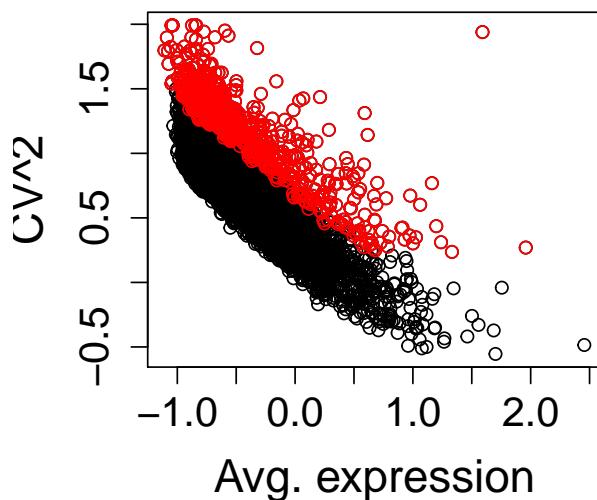
```
par(mfrow = c(3,2))
f<-0.1
n.hvg<-floor(f*length(Select))
hvg.deseq<-find.high.var.dm(data=NormDeseq$counts,n.hvg =n.hvg,
                               plot = TRUE,
                               title = "DESeq")
```

DESeq

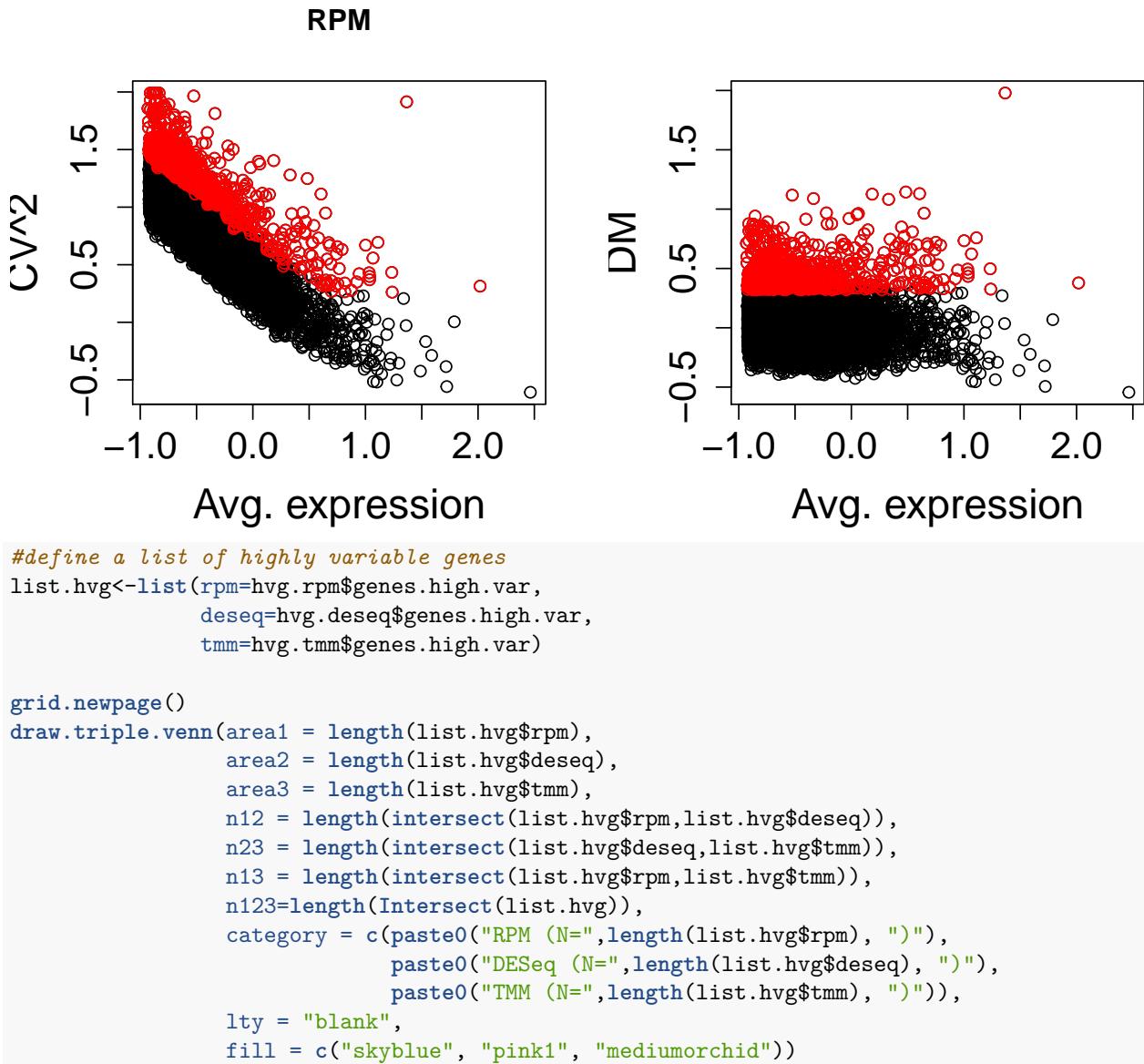


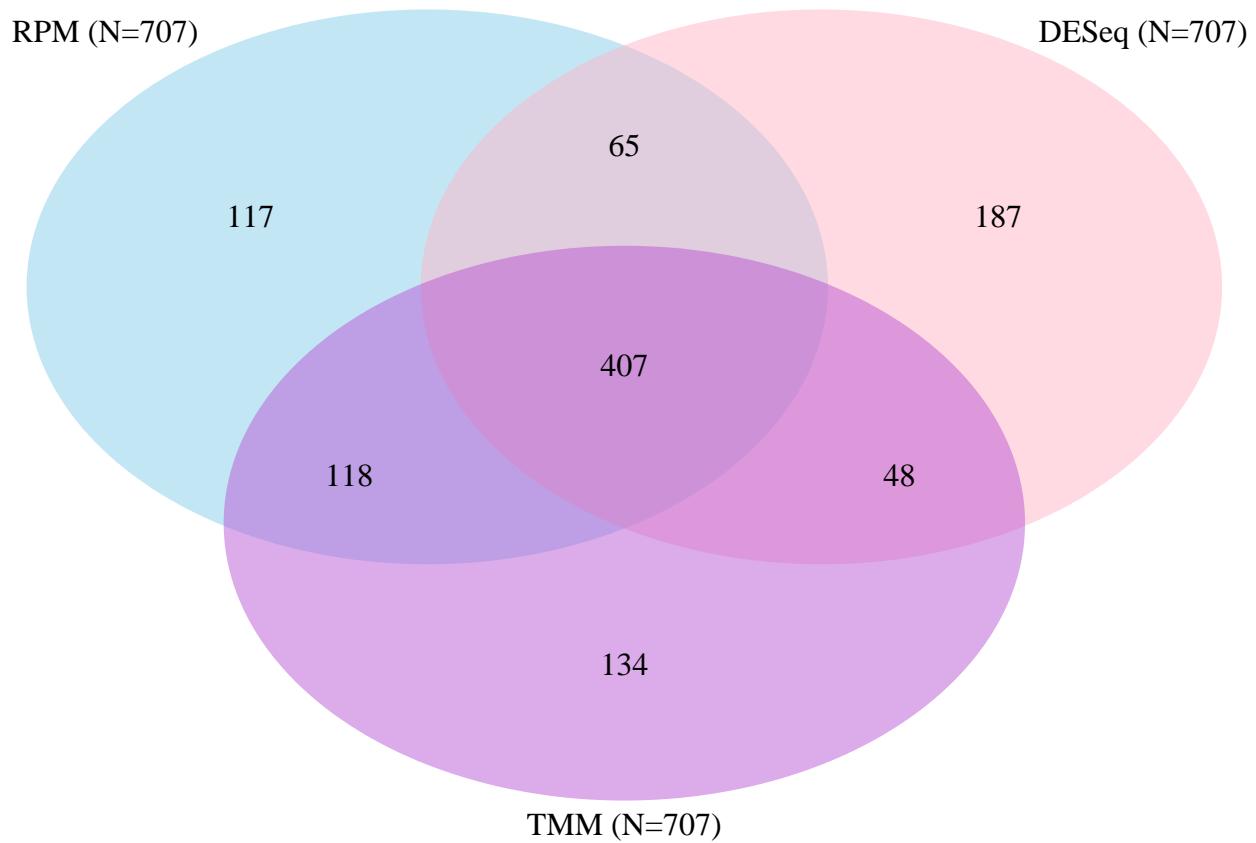
```
hvg.tmm<-find.high.var.dm(data=NormTmm$counts,n.hvg =n.hvg,  
                             plot = TRUE,  
                             title = "TMM")
```

TMM



```
hvg.rpm<-find.high.var.dm(data=NormRpm$counts,n.hvg =n.hvg,  
                            plot = TRUE,  
                            title = "RPM")
```





```
## (polygon[GRID.polygon.91], polygon[GRID.polygon.92], polygon[GRID.polygon.93], polygon[GRID.polygon.94])
407 out of 707 highly variable genes (58%) are shared across the three normalization methods.
```

Appendix

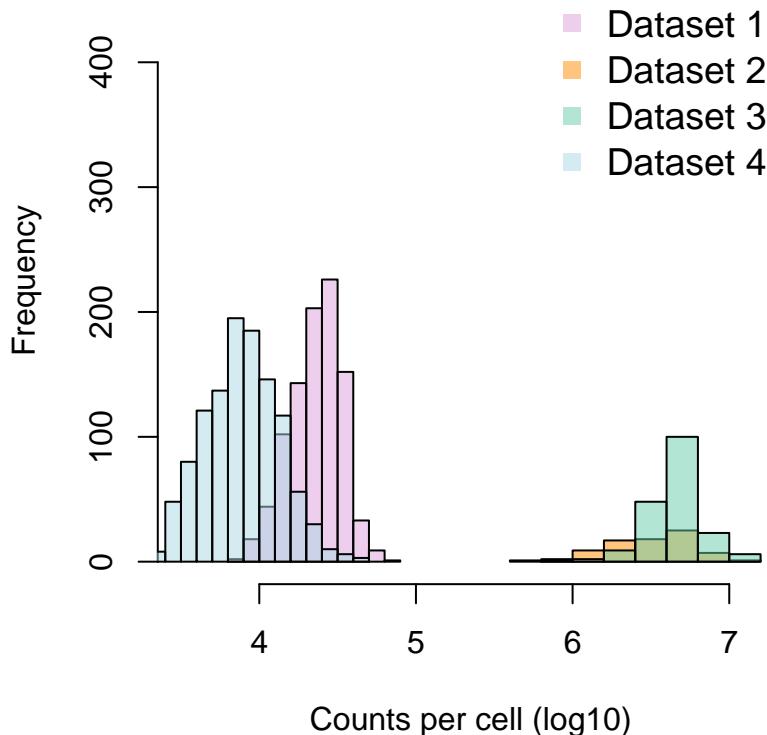
Comparison of global properties of the datasets

```
RawCounts<-zei[,c(idx.micro,idx.oligo,idx.astro)]
filter<-filter.genes(RawCounts)
Select<-filter$Select
RawCountsZeiFilter <- RawCounts[Select,]

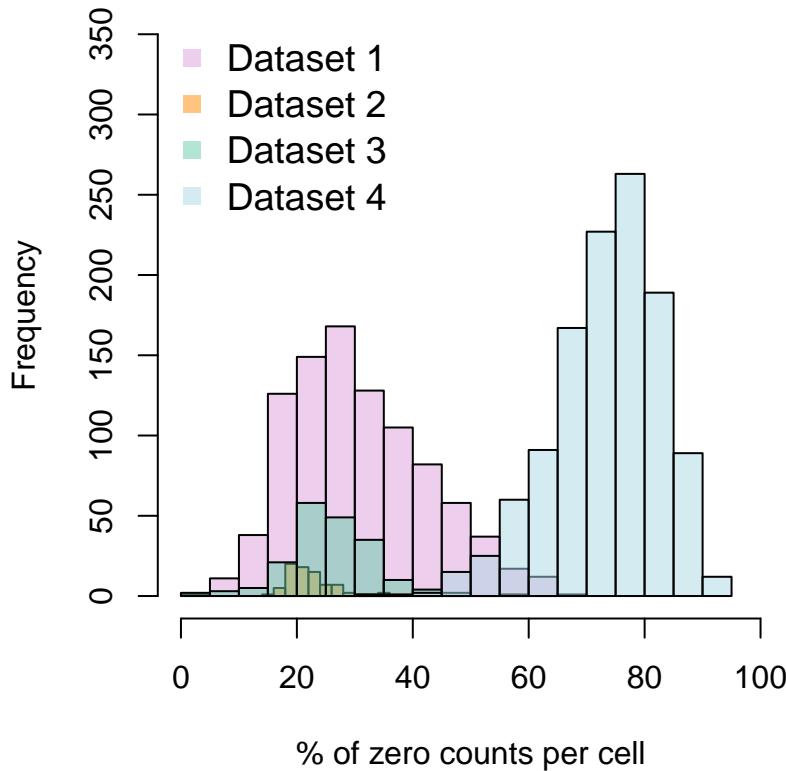
col1 <- rgb(221/255, 160/255, 221/255, 0.5)
col2 <- rgb(255/255, 140/255, 0/255, 0.5)
col3 <- rgb(102/255, 205/255, 170/255, 0.5)
col4 <- rgb(173/255, 216/255, 230/255, 0.5)

par(mfrow = c(1,1))
hist(log10(colSums(RawCountsOlaFilter)), col = col2,
     ylim=c(0,450), xlim = c(3.5, 7.2),
     xlab = "Counts per cell (log10)", main="")
hist(log10(colSums(RawCountsDropFilter)), col = col1, add = T)
hist(log10(colSums(RawCountsAllenFilter)), col = col3, add = T)
```

```
hist(log10(colSums(RawCountsZeiFilter)), col = col4, add = T)
legend('topright', c("Dataset 1", "Dataset 2", "Dataset 3", "Dataset 4"), bty = "n",
       pch = 15, col = c(col1, col2, col3, col4), cex = 1.2)
```



```
par(mfrow = c(1,1))
hist(100 * colMeans(RawCountsDropFilter == 0), col = col1,
     ylim=c(0,350), xlim = c(0, 100), main="",
     xlab = "% of zero counts per cell")
hist(100 * colMeans(RawCountsOlaFilter == 0), col = col2, add = T)
hist(100 * colMeans(RawCountsAllenFilter == 0), col = col3, add = T)
hist(100 * colMeans(RawCountsZeiFilter == 0), col = col4, add = T)
legend('topleft', c("Dataset 1", "Dataset 2", "Dataset 3", "Dataset 4"), bty = "n",
       pch = 15, col = c(col1, col2, col3, col4), cex = 1.2)
```



Function definitions

We define here some functions that have been used for the analysis.

```

library(data.table)
library(VennDiagram)
library(zoo)
library(statmod)
library(DESeq2)
library(edgeR)

## normalization methods
scran_norm <- function(x) {
  s <- computeSumFactors(x, sizes = c(20, 40, 60, 80, 100))
  counts <- t(t(x)/s*mean(s))
  n <- ncol(x)
  return(list(s=n*s/sum(s), counts=counts))
}

rpm <- function(x) {
  s <- colSums(x)
  counts <- t(t(x)/colSums(x)*mean(s))
  n <- ncol(x)
  return(list(s=n*s/sum(s), counts=counts))
}

deseq <- function(x) {
  s <- estimateSizeFactorsForMatrix(x)
}

```

```

counts <- t(t(x)/s*mean(s))
n <- ncol(x)
return(list(s=n*s/sum(s), counts=counts))
}

tmm <- function(x) {
  s <- calcNormFactors(x)*colSums(x)
  counts <- t(t(x)/s*mean(s))
  n <- ncol(x)
  return(list(s=n*s/sum(s), counts=counts))
}

# gene filtering
filter.genes<-function(raw.counts){

  raw.counts<-RawCounts

  #remove genes that are never expressed
  keep<-(rowSums(raw.counts)>0)
  raw.counts<-raw.counts[keep,]
  Rpm<-apply(raw.counts, 2, function(x) 1e6*(x/sum(x)))    #rpm(RawCounts)$counts
  Mean<-rowMeans(Rpm)
  Select<-names(Mean)[which(Mean>median(Mean))]

  return(list(Select=Select, Mean=Mean))
}

# hug
find.high.var.dm<-function(data,#matrix with normalized data
                           n.hvg, #the n.hvg genes with highest dm will be selected as hug
                           plot=TRUE,
                           title){

  df<-data.frame(log.avg=log10(apply(data, 1, function(x) mean(x))),
                 log.cv2=log10(apply(data, 1, function(x) (sd(x)/mean(x))^2)))
  )

  df.roll<-data.frame(roll.avg=rollapply(df[order(df[, "log.avg"]], decreasing=F), "log.avg",
                                         width=100, FUN=median, by=50),
                      roll.cv2=rollapply(df[order(df[, "log.avg"]],decreasing=F), "log.cv2",
                                         width=100, FUN=median, by=50))

  dm<-apply(df, 1,function(x){

    exp<-x["log.avg"]

    test<-which(df.roll[, "roll.avg"]>exp)
    if(length(test)==0) window=nrow(df.roll)
    else window=min(test)
  })
}

```

```

    return(x["log.cv2"]-df.roll[window,"roll.cv2"])

})

#select highly variable

high.var<-order(dm, decreasing=T)[1:n.hvg]

if(plot==TRUE){
  par(mfrow=c(1,2))
  plot(df[, "log.avg"],
        df[, "log.cv2"], xlab = "Avg. expression",
        ylab="CV^2", cex.lab=1.7, cex.axis=1.5, main=title)
  points(df[high.var, "log.avg"], df[high.var, "log.cv2"], col="red")

  plot(df[, "log.avg"], dm, xlab="Avg. expression",
        ylab="DM", cex.lab=1.7, cex.axis=1.5)
  points(df[high.var, "log.avg"], dm[high.var], col="red")
}

return(list(dm=dm, genes.high.var=high.var))

}

#plotting functions
plot.sf.ratio.frac.zeros.cata<-function(df, ...)
{
  args = list(...)
  if("ylim.max" %in% names(args)) {ylim.aux = args$ylim.max}
  else{ylim.aux = max(df$y / df$x, df$z / df$x)}

  par(mfrow = c(1,2))
  par(cex.lab = 1.7, cex.axis = 1.5)
  par(mar = c(7, 7, 4, 4) + 0.1)
  smoothScatter(100*df$zeros, df$y / df$x,
                ylim = c(0, ylim.aux),
                xlab = "% of zeros per cell",
                ylab = "Ratio scaling factors (DESeq/RPM)")
  abline(h=1, lty=2, col = "red", lwd = 3)
  smoothScatter(100*df$zeros, df$z / df$x,
                ylim = c(0, ylim.aux),
                xlab = "% of zeros per cell",
                ylab = "Ratio scaling factors (TMM/RPM)")
  abline(h=1, lty=2, col = "red", lwd = 3)

}

```

```

# #Multiple set version of intersect, union and setdiff
Intersect <- function (x) { #x is a list

  if (length(x) == 1) {
    unlist(x)
  } else if (length(x) == 2) {
    intersect(x[[1]], x[[2]])
  } else if (length(x) > 2){
    intersect(x[[1]], Intersect(x[-1]))
  }
}

pairs.comparison.diff <-function(df1, df2, title,
                                Norm.Colour = c("lightpink3", "darkolivegreen3", "darkgoldenrod1"))
{

  df1<-log10(df1)
  df2<-log10(df2)

  diff.df1<-data.frame(x=df1[,1]-df1[,2],y=df1[,2]-df1[,3],z=df1[,1]-df1[,3])
  diff.df2<-data.frame(x=(df2[,1])-(df2[,2]),y=df2[,2]-df2[,3],z=df2[,1]-df2[,3])

  avg.df1<-data.frame(x=rowMeans(df1[,1:2]),
                        y=rowMeans(df1[,2:3]),
                        z=rowMeans(df1[,c(1,3)]))
  avg.df2<-data.frame(x=rowMeans(df2[,1:2]),
                        y=rowMeans(df2[,2:3]),
                        z=rowMeans(df2[,c(1,3)]))

  ylim1<-c(min(diff.df1), max(diff.df1))
  ylim2<-c(min(diff.df2), max(diff.df2))

  xlim1<-c(min(avg.df1), max(avg.df1))
  xlim2<-c(min(avg.df2), max(avg.df2))

  par(mfrow = c(3,3))
  par(mar = c(7, 10, 4, 4) + 0.1, oma=c(0,0,3,0))
  par(mgp = c(5, 2, 0))
  par(cex.lab = 3, cex.axis = 2)

  plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n',
       xaxt = 'n', yaxt = 'n', bg = Norm.Colour[1])
  rect(0, 0, 1, 1, col = Norm.Colour[1])
  text(x = 0.5, y = 0.5, paste("RPM"), cex = 8, col = "white", lwd = 2)

  smoothScatter(avg.df1$x, diff.df1$x,
                ylab = "Scaling factor\n RPM/DESeq [log10] ",
                xlab = "Avg. Log10(scaling factor)", ylim=ylim1,xlim=xlim1)#
  abline(h = 0, lty = 2, lwd = 3, col = "red")
}

```

```

smoothScatter(avg.df1$z, diff.df1$z,
              ylab = "Scaling factor\n RPM/TMM [log10] ",
              xlab = "Avg. Log10(scaling factor)", ylim=ylim1, xlim=xlim1)#
abline(h = 0, lty = 2, lwd = 3, col = "red")

smoothScatter(avg.df2$x, diff.df2$x, #log10
              xlab = "Avg. Log10(CV^2)",
              ylab = "CV^2\n RPM/DESeq [log10]", ylim=ylim2, xlim=xlim2)#
abline(h = 0, lty = 2, lwd = 3, col = "red")

plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n',
      xaxt = 'n', yaxt = 'n', bg = Norm.Colour[2])
rect(0, 0, 1, 1, col = Norm.Colour[2])
text(x = 0.5, y = 0.5, paste("DESeq"), cex = 8, col = "white", lwd = 2)

smoothScatter(avg.df1$y, diff.df1$y,
              ylab = "Scaling factor\n DESeq/TMM [log10] ",
              xlab = "Avg. Log10(scaling factor)", ylim=ylim1, xlim=xlim1)#
abline(h = 0, lty = 2, lwd = 3, col = "red")

smoothScatter(avg.df2$z, diff.df2$z,
              xlab = "Avg. Log10(CV^2)",
              ylab = "CV^2\n RPM/TMM [log10]", ylim=ylim2, xlim=xlim2)#
abline(h = 0, lty = 2, lwd = 3, col = "red")

smoothScatter((avg.df2$y), (diff.df2$y), #log10
              xlab = "Avg. Log10(CV^2)",
              ylab = "CV^2\n DESeq/TMM [log10]", ylim=ylim2, xlim=xlim2)#
abline(h = 0, lty = 2, lwd = 3, col = "red")

plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n',
      xaxt = 'n', yaxt = 'n', bg = Norm.Colour[3])
rect(0, 0, 1, 1, col = Norm.Colour[3])
text(x = 0.5, y = 0.5, paste("TMM"), cex = 8, col = "white", lwd = 2)

title(title, outer=TRUE, cex.main = 3)
}

```

References

Klein, Allon M, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A Weitz, and Marc W Kirschner. 2015. “Droplet Barcoding for Single-Cell Transcriptomics Applied to Embryonic Stem Cells.” *Cell* 161 (5). Elsevier: 1187–1201.

Kolodziejczyk, Aleksandra A, Jong Kyoung Kim, Jason C H Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N Natarajan, Alex C Tuck, et al. 2015. “Single Cell RNA-Sequencing of Pluripotent States Unlocks Modular Transcriptional Variation.” *Cell Stem Cell* 17 (4): 471–85.

Tasic, Bosiljka, Vilas Menon, Thuc Nghi Nguyen, Tae Kyung Kim, Tim Jarsky, Zizhen Yao, Boaz Levi, et al. 2016. “Adult Mouse Cortical Cell Taxonomy Revealed by Single Cell Transcriptomics.” *Nature Neuroscience* 19 (2). Nature Publishing Group: 335–46.

Zeisel, Amit, Ana B Muñoz-Manchado, Simone Codeluppi, Peter Lönnerberg, Gioele La Manno, Anna Juréus,

Sueli Marques, et al. 2015. “Cell Types in the Mouse Cortex and Hippocampus Revealed by Single-Cell RNA-Seq.” *Science* 347 (6226). American Association for the Advancement of Science: 1138–42.